НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ
КАФЕДРА МАТЕМАТИЧНИХ МЕТОДІВ ЗАХИСТУ ІНФОРМАЦІЇ

«До захисту допущено»
В.о. завідувача кафедрою
_____ М.М.Савчук
(підпис) (ініціали, прізвище)
“___”_____20 _ р.

Дипломна робота

на здобуття ступеня бакалавра

з напряму підготовки : 113 «Прикладна математика»
(код і назва)

на тему: Швидкі кореляційні атаки на малоресурсні Grain - подібні потокові шифри

Виконав (-ла): студент (-ка) __4__ курсу, групи _ФІ-62_____
(шифр групи)

Кузнєцов Павло Сергійович
(прізвище, ім’я, по батькові) (підпис)

Керівник професор кафедри ММЗІ, д-р. техн. наук. Олексійчук А.М.
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище, ініціали) (підпис)

Рецензент
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій дипломній роботі
немає запозичень з праць інших авторів
без відповідних посилань.
Студент _____
(підпис)

Київ – 2020року

Національний технічний університет України
«Київський політехнічний інститут
імені Ігоря Сікорського»
Фізико-технічний інститут

Кафедра математичних методів захисту інформації

Рівень вищої освіти – перший (бакалаврський)

Напрям підготовки - 113 «Прикладна математика»

ЗАТВЕРДЖУЮ
В.о. завідувача кафедрою

М.М.Савчук

_____
(підпис) (ініціали, прізвище)

«___»_____20__ р.

ЗАВДАННЯ
на дипломну роботу студенту
Кузнєцову Павлу Сергійовичу

(прізвище, ім'я, по батькові)

1. Тема роботи Швидкі кореляційні атаки на малоресурсні Grain - подібні потокові шифри ,

керівник роботи професор кафедри ММЗІ,
д-р. техн. наук. Олексійчук А.М. ,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від _____ р. № _____

2. Термін подання студентом роботи

3. Вихідні дані до роботи розроблено міні версію шифру Grain-v0, що буде використовуватись у подальшому аналізі швидких кореляційних атак на шифри сімейства Grain.

4. Зміст роботи У цій роботі зроблено огляд теоретичних матеріалів, що містять у собі аналіз структури шифрів сімейства Grain, а також швидкої кореляційної атаки. Запропоновано міні версію шифру Grain-v0.

5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо) Презентація

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1	провести огляд опублікованих джерел за тематикою дослідження		
2	робити детальний аналіз загальної структури шифру Grain		
3	робити детальний аналіз швидкої кореляційної атаки		
4	розробити міні версію шифру Grain-v0		
5	написати програму, яка реалізовує міні версію шифру Grain-v0		

Студент _____ _Кузнєцов П.С_____
(підпис) (ініціали, прізвище)

Керівник роботи _____ _Олексійчук А.М_____
(підпис) (ініціали, прізвище)

РЕФЕРАТ

Кваліфікаційна робота містить: 65 стор., 10 рисунків, 3 таблиці, 14 джерел та 1 додаток.

У цій роботі зроблен детальний аналіз існуючої швидкої кореляційної атаки на першу версію потокового шифру Grain, а саме Grain-v0 та теоретичних матеріалів, що потрібні для її розуміння. Розроблено міні версію шифру Grain-v0.

Тема роботи: швидкі кореляційні атаки на малоресурсні Grain - подібні потокові шифри.

Мета роботи: дослідження властивостей шифрів сім'ї Grain, що впливають на їх стійкість відносно швидких кореляційних атак.

Задача роботи: створення міні версії шифру Grain-v0.

Об'єкт дослідження: процес перетворення інформації у Grain-подібних потокових шифрах.

Предмет дослідження: властивості компонентів шифрів сімейства Grain.

Методи дослідження: методи математичної статистики, теорії булевих функцій, абстрактної алгебри.

У результаті роботи цієї роботи запропоновано міні версію шифру Grain-v0, що буде використовуватись для подальшого дослідження шифрів сімейства Grain, а саме, швидких кореляційних атак. Також представлена реалізація шифру Grain-v0 та його міні версії.

ПОТОКОВІ ШИФРИ, СИМЕТРИЧНА КРИПТОГРАФІЯ, GRAIN, GRAIN-V0

ABSTRACT

The thesis contains: 65 pages, 10 figures, 3 tables, 14 sources and 1 appendix.

In this thesis, a detailed analysis of the existing fast correlation attack on the first version of the Grain stream cipher, namely Grain-v0, and the theoretical materials needed for it understanding. Developed a mini version of the cipher Grain-v0.

The theme of this thesis is fast correlation attacks on low-resource Grain - similar stream ciphers.

The goal of this thesis is to study the structure of ciphers of the Grain family, and fast correlation attack.

The task of this work is to create a mini version of the cipher Grain-v0.

The subject of the research are properties of cipher components of the Grain family.

Methods of research are methods of mathematical statistics, theory of boolean functions, abstract algebra.

As a result of this work, a mini version of the cipher Grain-v0 was proposed, which will be used for further study of ciphers of the Grain family, namely, fast correlation attacks. The implementation of the Grain-v0 cipher and its mini-version is also presented.

STREAM CIPHERS, SYMMETRIC CRYPTOGRAPHY, GRAIN, GRAIN-V0

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

$\oplus$ - операція додавання за модулем 2

РЗЛЗЗ - регістр зсуву з лінійним зворотним зв'язком.

РЗНЗЗ - регістр зсуву з нелінійним зворотним зв'язком.

РЗЗЗ - регістр зсуву зі зворотнім зв'язком.

Grain - сімейство шифрів симетричного потокового шифрування, що орієнтовані на апаратну реалізацію.

ШПУ - швидке перетворення Уолша

ШКА - швидка кореляційна атака

ВСТУП

Актуальність дослідження. У сучасному світі стрімливо розвивається інтернет речей, а разом з ним безпека інтернету речей, де криптографія відіграє важливу роль. Деякі пристрої забезпечені доволі потужними процессорами, що можуть використовувати алгоритми для звичайних комп'ютерів, однак більшість забезпечена малопотужними процессорами, що можуть лише невелику частину своїх обчислювальних потужностей виділити для безпеки. Оскільки пристрої за розміром малі, вони використовують малі батареї, або самі породжують енергію. Тому криптографічні алгоритми, які потрібно використовувати в таких випадках повинні бути "малими".

Типовим прикладом використання "легковагої"криптографії є мітки радіочастотної ідентифікації. Вони використовуються для ідентифікації у громадському транспорті, контролі за переміщенням персоналу, системах електронних платежів, біометричних паспортах та інших галузях.

Для апаратної реалізації симетричних потокових шифрів було запропоновано сімейство шифрів Grain. Яке є актуальним матеріалом для досліджень.

У цій роботі досліджується швидка кореляційна атака на Grain-v1, а також пропонується міні версія шифру Grain-v0.

Метою дослідження є дослідження властивостей шифрів сімейства Grain, що впливають на їх стійкість відносно швидких кореляційних атак. Для досягнення мети необхідно розв'язати **задачу дослідження**, яка полягає у створенні міні версії шифру Grain-v0. Для розв'язання задачі необхідно вирішити такі завдання:

1) провести огляд опублікованих джерел за тематикою дослідження;
2) зробити детальний аналіз загальної структури шифру Grain;
3) зробити детальний аналіз швидкої кореляційної атаки;
4) розробити міні версію шифру Grain-v0.

5) написати програму, яка реалізовує міні версію шифру Grain-v0.

Об'єктом дослідження є процес перетворення інформації у Grain-подібних потокових шифрах.

Предметом дослідження є властивості компонентів шифрів сімейства Grain.

При розв'язанні поставлених завдань використовувались такі *методи дослідження* : методи математичної статистики, теорії булевих функцій, абстрактної алгебри. Для програмної реалізації було використано мову програмування python.

Практичне значення результатів полягає у використанні міні версії шифру Grain-v0 для подальших досліджень швидких кореляційних атак на сімейство шифрів Grain.

1 СТРУКТУРА ШИФРУ GRAIN-V0

У данному розділі розглядаються необхідні теоритичні відомості для опису шифрів сімейства Grain-v0. А саме опис у стандартних термінах потокових шифрів, та скінченних автоматів.

1.1 Загальна структура потокових шифрів. Основні терміни.

Потоковий шифр - це криптосистема, що складається з двох частин, а саме: генератору гамми, та функції виходу. Для роботи генератору гамми його требя проініціалізувати, визначити спосіб вироблення символу гамми, та оновлення стану. Функція виходу на вхід приймає символ відкритого тексту, символ гамми, та додаткову інформацію. Зазвичай функцію виходу обирають $Out(x_i,z_i,R_i) = x_i \oplus z_i$

Потокові шифри бувають синхронні та із самосинхронізацією. Потік гамми у синхронних шифрах не залежить від відкритого тексту та шифротексту, тоді як потік гамми у самосинхронізованих залежить від n попередніх символів шифротексту.

Генератор гамми із самосинхронізацією:

1) $Init(K,\, IV) \rightarrow$ S, c_0, c_{-1}, c_{-2},...,c_{-n+1}
2) $z_i = Stream(S,\, K,\, c_{i-1},\, c_{i-2},...,c_{i-n})$
3) $c_i = Out(x_i,\, z_i,\, R_i)$

Синхронний генератор гамми: (автомат без пам'яті)

1) $S_0 = Init(K,\, IV)$
2) $z_i = Stream(K,\, S_i)$
3) $S_{i+1} = Next(K,\, S_i)$

1.2 Структура шифру Grain-v0

Шифр Grain є синхронним потоковим шифром [1], який був запропонований Мартином Хеллом, Томасом Йоханссоном та Віллі Мейером у 2004 році. Алгоритм став одним з фіналістів у конкурсі eSTREAM для апаратної реалізації.

Шифр складається складається з трьох основних частин, а саме: РЗЛЗЗ, РЗНЗЗ, та функції фільтрації. Стан РЗЛЗЗ позначимо через s_i,s_{i+1},...,s_{i+79}, а стан РЗНЗЗ позначимо через b_i,b_{i+1},...,b_{i+79}.

Поліном зворотнього зв'язку для РЗЛЗЗ визначається як $f(x) = 1 + x^{18} + x^{29} + x^{42} + x^{57} + x^{67} + x^{80}$ - незвідний поліном ступеня 80.

Для зворотнього зв'язку регістру РЗНЗЗ визначається поліном $g(x) = 1+x^{17}+x^{20}+x^{28}+x^{35}+x^{43}+x^{47}+x^{52}+x^{59}+x^{65}+x^{71}+x^{80}+x^{17}x^{20}+x^{43}x^{47}$ $+ x^{65}x^{71} + x^{20}x^{28}x^{35} + x^{47}x^{52}x^{59} + x^{17}x^{35}x^{52}x^{71} + x^{20}x^{28}x^{43}x^{47} + x^{17}x^{20}x^{59}x^{65}$ $+ x^{17}x^{20}x^{28}x^{35}x^{43} + x^{47}x^{52}x^{59}x^{65}x^{71} + x^{28}x^{35}x^{43}x^{47}x^{52}x^{59}$

Для бітів регістру РЗЛЗЗ отримаємо вираз $s_{i+80} = s_{i+62} + s_{i+51}$ $+ s_{i+38} + s_{i+23} + s_{i+13} + s_i$.

А для бітів регістру РЗНЗЗ отримаємо вираз $b_{i+80} = s_i+b_{i+63}+b_{i+60}+b_{i+52}+b_{i+45}+b_{i+37}+b_{i+33}+b_{i+28}+b_{i+21}+b_{i+15}+b_{i+9}+b_i+$ $b_{i+63}b_{i+60}+b_{i+37}b_{i+33}+b_{i+15}b_{i+9}+b_{i+60}b_{i+52}b_{i+45}+b_{i+33}b_{i+28}b_{i+21}+b_{i+63}b_{i+45}b_{i+28}b_{i+9}+$ $b_{i+60}b_{i+52}b_{i+37}b_{i+33} + b_{i+63}b_{i+60}b_{i+21}b_{i+15} + b_{i+63}b_{i+60}b_{i+52}b_{i+45}b_{i+37} +$ $b_{i+33}b_{i+28}b_{i+21}b_{i+15}b_{i+9} + b_{i+52}b_{i+45}b_{i+37}b_{i+33}b_{i+28}b_{i+21}$.

Стан шифру описується вмістом регістрів зсуву. З регістрів обирається 5 бітів, які подаються на вхід булевої функції $h(x)$, яка є збалансованою ($wt(h) = 2^{5-1}$, де wt - вага булевої функції), має кореляційний імунітет першого порядку, та алгебраїчний степінь 3. Функція h має максимальну нелінійність, а саме 12. $NL_h = \min_{g \in A_n} dist(h,g)$. Де $dist(h,g) = wt(h \oplus g)$. Біти беруться на вхід як з РЗЛЗЗ так і з РЗНЗЗ. Визначається функція як $h(x) = x_1+x_4+x_0x_3+x_2x_3+x_3x_4+x_0x_1x_2+x_0x_2x_3+x_0x_2x_4+x_1x_2x_4+x_2x_3x_4$.

Змінні функції x_0, x_1, x_2, x_3 та x_4 відповідають позиціям бітів s_{i+3}, s_{i+25}, s_{i+46}, s_{i+64} та b_{i+63}. Результуючий біт маскується бітом b_i з РЗНЗЗ.

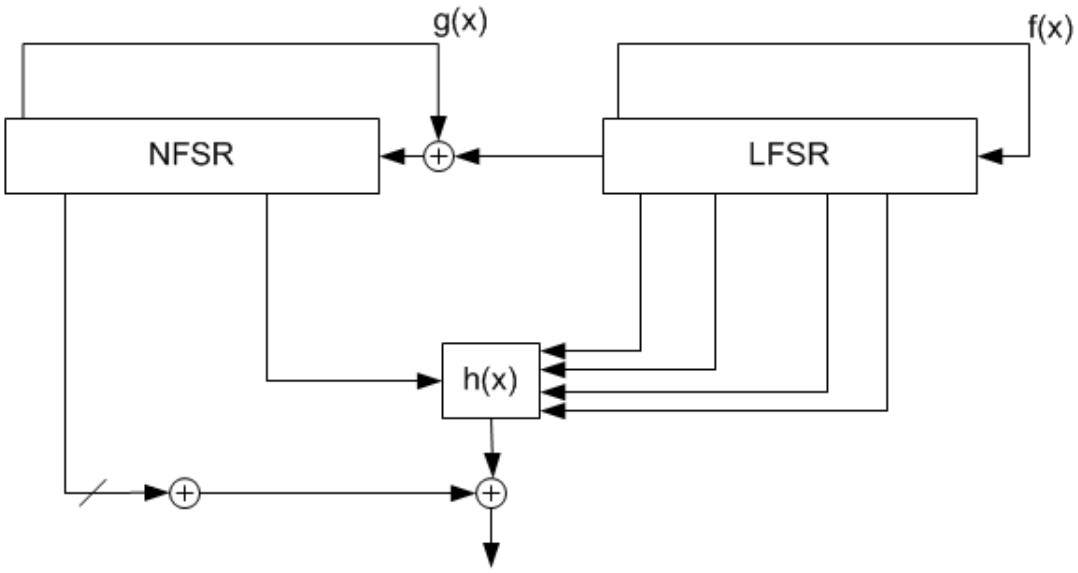

Рисунок 1.1 – Схема роботи шифру Grain-v0

1.3 Ініціалізація та робота шифру Grain

Перед початком роботи шифр треба проініціалізувати, для цього на вхід подаються ключ $K = k_0,...,k_{79}$ та вектор ініціалізації $IV = iv_0,...,iv_{63}$. Спочатку РЗНЗЗ заповнюється 80 бітовим ключем $b_i = k_i$, $0 \leq i \leq 79$, потім заповнюємо перші 64 біти регістру РЗЛЗЗ вектором ініціалізації $s_i = iv_i$, $0 \leq i \leq 63$, інші біти заповнюємо одиницями $s_i = 1$, $64 \leq i \leq 79$. Це потрібно для того, щоб регістри не заповнювались нулями. Для завершення ініціалізації виконується 160 тактів без генерації гамми. Замість подачі біту гамми на вихід він ксориться з результатом зворотніх функцій обох регістрів, та стає частиною оновлення їх стану. Після виконання 160 тактів можна генерувати гамму.

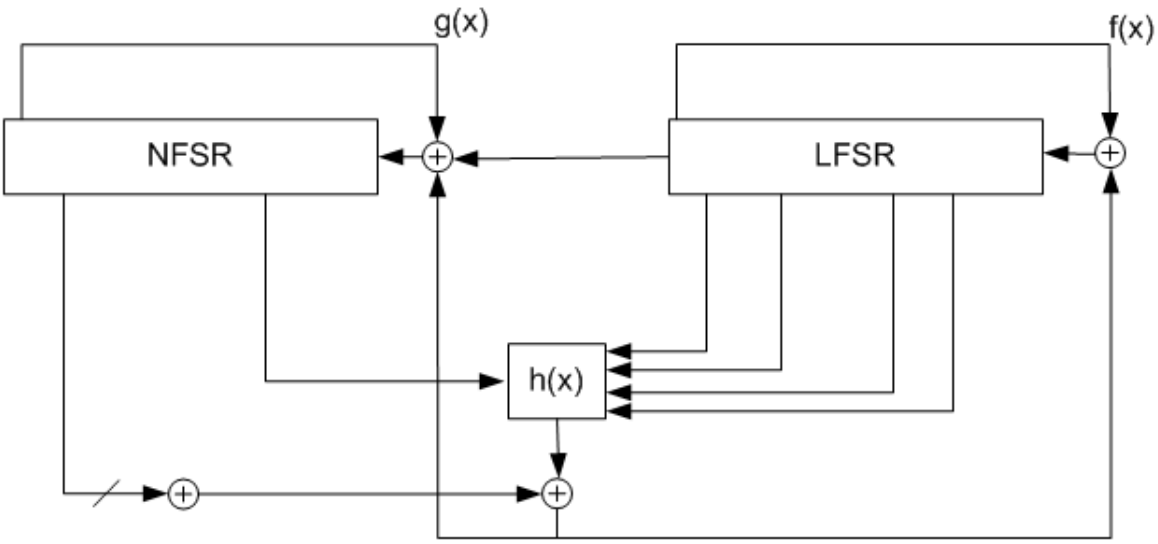

Рисунок 1.2 – Ініціалізація шифру Grain-v0

1.4 Особливості побудови та прискорення роботи

Алгоритм розроблений таким чином щоб його було легко реалізовувати на апаратних пристроях. Вимогою до подібних апаратних шифрів є безпека, що відповідає обчислювальній складності 2^{80}, а саме таких результатів можна досягти завдяки пам'яті в 160 біт. Однак на безбеку також впливають функції, що оперують бітами. Вони не повинні бути занадто великими, оскільки чим більше функції, тим більше вентилів вони потребують, однак і дуже малими вони також бути не можуть.

Доречним буде спостереження, що РЗЛЗЗ має примітивній поліном зворотнього зв'язку, це означає, що регістр генерує m - послідовність. Яка має максимальний період $2^{80} - 1$, та біти 0 та 1 зустрічаються майже з однаковою ймовірністю. Тобто кількість нулів $= 2^{n-1} - 1$, а кількість одиниць $= 2^{n-1}$. Такий регістр ще називають полноцикловим.

Можна помітити, що РЗНЗЗ використовує біт РЗЛЗЗ для оновлення стану, ця деталь була розроблена, щоб регістр РЗНЗЗ був збалансований, на цю властивість ще впливає біт b_{i+79}. Також вона робить період вихідної послідовності залежним від ключа, та вектора ініціалізації.

Як було описано в попередніх пунктах, перед генерацією гамми треба зробити 160 тактів. Це зроблено з метою улаштувати неочікуваний стан регістрів. Однак при регулярній зміні ключа та вектора ініціалізації виникає проблема в постійному виконанні 160 тактів, це впливає на час передачі інформації. Тому кількість тактів обрана спираючись на компроміс між безпекою та швидкістю.

Цікавою властивістю є відсутність використання бітів b_i, $65 \leq i \leq 79$, та s_i, $65 \leq i \leq 79$ у функціях зворотнього зв'язку та функції фільтрації. Це дозволяє пришвидшити алгоритм за допомогою використання зворотніх функцій $f(x)$, $g(x)$, а також $h(x)$ одразу у декількох місцях. Кількість місць для встановлення функцій обмежена, а саме ≤ 16. Якщо t - це кількість біт які буде генерувати шифр за один такт, то $160/t$ - кількість тактів які потрібно буде зробити для ініціалізації перед початком роботи.

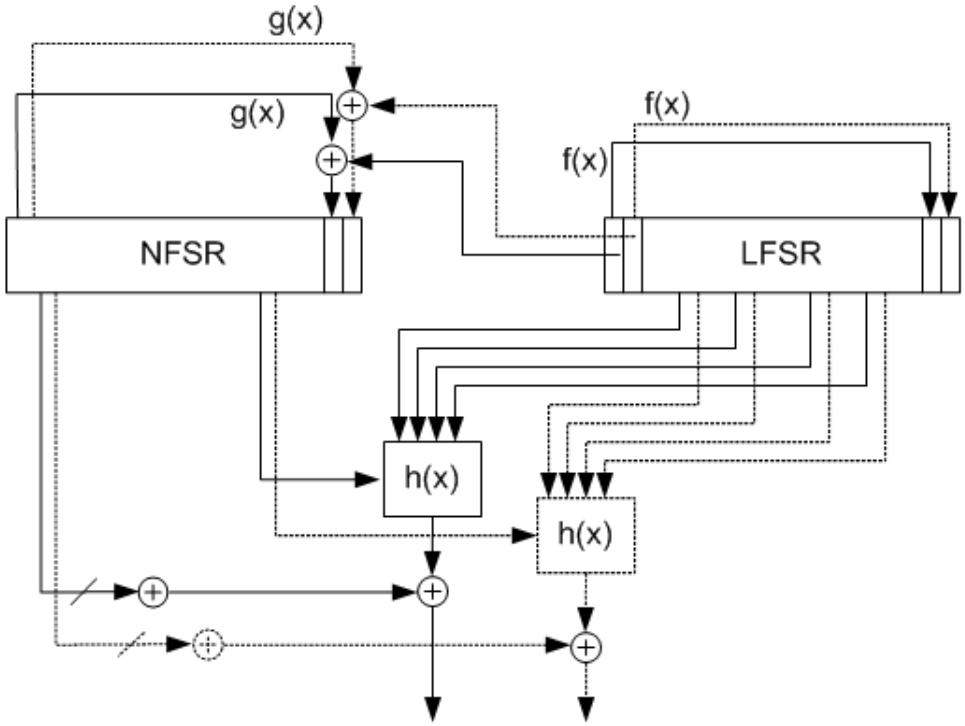

Рисунок 1.3 – Прискорення шифру Grain-v0

1.5 Криптоаналіз шифру Grain

При створенні нового криптоалгоритму його одразу перевіряють на стійкість до стандартних атак. Атака не повинна бути швидшою за звичайний підбір ключа. Тобто не швидше, аніж $\mathcal{O}(2^{80})$

1.5.1 Кореляційний криптоаналіз

Біти регістру РЗЛЗЗ є збалансованими, однак ця властивість не розповсюджується на РЗНЗЗ, лише завдяки біту з РЗЛЗЗ, що ксориться з функцією зворотнього зв'язку $g(x)$ РЗНЗЗ стає забалансованим. Крім того пригадаємо, що $g(x)$ також є збалансованою функцією. З цих властивостей можемо припустити, що ці два регістри не корелюють один з одним. Функція $h(x)$ має кореляційний імунітет першого порядку, попри все це не означає, що не існує кореляцій між результатом функції та входами. $h(x)$ приймає на вхід 5 змінних, одна з яких належить РЗНЗЗ, а вихід ксориться з бітом b_i, що також належить РЗНЗЗ. Наразі кореляції між результуючими бітами та бітами РЗЛЗЗ дуже малі, тому атаки швидкої кореляції потребують подальших досліджень.

1.5.2 Алгебраїчні атаки

Функція $h(x)$ має алгебраїчний степінь 3, яка є дуже нестійкою до алгебраїчних атак, з іншої точки зору така атака не зможе відновити стан у 160 біт, причиною тому є доволі велкий алгебраїчний степінь нелінійної функції зворотнього зв'язку РЗНЗЗ. Припустимо, що ми можемо представити результат генератору лише як функцію від бітів РЗЛЗЗ, за такої умови алгебраїчний степінь вихідних бітів буде великим, та змінюватиметься з часом. Це спричиняє протистояння алгебраїчним атакам.

1.5.3 Атака компромісу часу/пам'яті/данних

Атаки компромісу часу/пам'яті/данних на потокові шифри коштують $\mathcal{O}(2^{n/2})$, де n це кількість внутрішніх станів шифру. В випадку Grain $n = 160$. Відомим є факт, що атаки компромісу для шифрів з низьким супротивом вибірки мають невеликі таблиці пошуку, та більший вибір параметрів. Наразі функція $h(x)$ має правильно підібраний супротив вибірки, при використанні меншої від 5 кількості змінних, а саме до 3, вона не стає лінійною. Аналогічно, змінні які потрапляють в одночлени функції $g(x)$ є достатньо розрізненими. З цих властивостей випливає, що загалом супротив вибірки буде достатньо великим, а атаки компромісу часу/пам'яті/данних будуть мати складність не нижче за $\mathcal{O}(2^{80})$.

1.5.4 Атака на основі підібраних векторів

Необхідною умовою для захисту від диференціальних або статистичних атак обраного вектору ініціалізації є те, що початкові стани для множини векторів ініціалізації алгебраїчно та статистично незалежні. Кількість тактів для ініціалізації підібрано саме так, щоб вага Хеммінга різниці повного, проініціалізованого 160 бітового стану для двох векторів після ініціалізації була близькою до випадкової. Цей засіб повинен протистояти атакам обраного вектору ініціалізації.

Оскільки постійне виконання ініціалізації потребує забагато часу, виникає потреба в зменшенні кількості тактів. Після 80 тактів стан шифру буде залежати як від ключа так і від вектору. Використовуючи таку атаку можна переініціалізувати шифр з таким самим ключем, але іншим вектором ініціалізації, який відрізняється від попереднього лише одним бітом. Як приклад, розглянемо випадок коли для ініціалізації алгоритму потрібно 80 тактів, та останній біт вектора ініціалізації iv_{63} інвертовано,

тобто s_{i+63} також інвертовано. Така ситуація може виникнути, якщо вектор IV обирається як порядковий номер. Аналізуючи різницю станів після ініціалізації, можна зробити спотереження, що деякі позиції доволі передбачувані. Біт s_{63} не використовується в зворотньому зв'язку або функції фільтрації, тому оновлення першого регістру буде однаковим в обоих випадках, біт s_0 буде однаковим в обох ініціалізаціях. Наступна позиція, яку займе інвертований біт буде s_{62}. Ця позиція використовується у лінійному зворотньому зв'язку регістру РЗЛЗЗ, тому біт s_1 завжди буде відрізнятись при іншій ініціалізації. Схожі методи можуть бути використані, для того щоб показати, що більше ніж половина станів може бути детерміновано. Ця детермінованість може використовуватись для атаки. Нехай $\underline{x} = x_0,x_1,x_2,x_3,x_4,x_5$ - змінні для функції $h(x)$ - після першої ініціалізації, та $\underline{x}_\delta = x_0,x_1,x_2,x_3,x_4,x_5$ - змінні для функції $h(x)$ - після другої ініціалізації. Тепер треба вирахувати $P(\underline{x},\underline{x}_\delta)$, якщо такий розподіл є зміщеним, то розподіл різниці в першому вихідному біті $P(h(\underline{x})h(\underline{x}_\delta))$ також зміщений. Припустимо, що $P(h(\underline{x})$ $\oplus \mathrm{h}(\underline{\mathrm{x}}_\delta) = 0) = 1/2 + \epsilon$. Тоді кількість ініціалізацій яка нам протрібна буде порядку $1/\epsilon^2$. Така атака може бути оптимізована за допомогою вирахування біта, який буде давати найбільше відхилення, оскільки не обов'язково, що біти в регістрах будуть відповідати вхідним бітам функції $h(x)$, що має детерміновану різницю після ініціалізації. Ця атака показує, що ймовірність того, що після ініціалізації з двома різними IV будь-який біт стану буде однаковий повинна бути близькою до 1/2. Як у випадку з 80 тактами у станах будуть однакові біти й у 96, 112 та 128 тактах.

1.5.5 Атака на основі помилок

Наразі серед найпотужніших атак можливих на будьякому шифрі є атаки помилок. Вони й наразі є ефективними на потокових шифрах, це в свою чергу ускладнює задачу проектування шифру та надання йому певного рівня безпеки.

План атаки базується на тому, що зловмисник може застосувати деякі помилкові інвертації бітів до одного з регістрів. Однак, він лише частково розпоряджається кількістю, місцем та точним часом. Більш сильне припущення, яке можна зробити, полягає в тому, що він здатний перевернути один біт, в певний момент часу та місці які йому не відомі. Зловмиснику також може бути дана можливість повернути пристрій у початковий стан, та використати ще одну помилкову атаку. Адаптуючи методи під данний алгоритм. Найсильнішим припущенням може бути те, що атакуючий якимось чином спровокує помилку в регістрі РЗЛЗЗ, та знайде відповідну позицію помилки, однак це лише прирущення. Метою криптоаналітика є дослідження входів та виходу функції фільтрації $h(x)$, та отримання інформації про 5 входів, аналізуючи вже відомі пари входів та виходів. Це схоже на аналіз S - блоків у симетричному шифрі DES. Допоки різниця спровокована помилкою в РЗЛЗЗ не розповсюджується на біт b_{i+63}, різниця, що спостерігається у виході шифру, надходить від входів $h(x)$ тільки від РЗЛЗЗ. Так само можно застосовувати атаку на РЗНЗЗ, та помилки не будуть впливати на РЗЛЗЗ, однак помилки в РЗНЗЗ розповсюджуються нелінійно та їх складніше предбачити, тому атака на РЗНЗЗ виглядає складнішою.

Висновки до розділу 1

У цьому розділі проаналізовані джерела та наведені поняття, щознадобляться для подальшої роботи, а саме описані шифр Grain-v0. Також наведений невеликий перелік криптографічних атак на шифри сімейства Grain, а саме: кореляційні атаки, алгебраїчні атаки, атаки компромісу часу, пам'яті та данних, атаки на основі підібраних векторів, та атаки на основі помилок.

2 ОГЛЯД ШВИДКОЇ КОРЕЛЯЦІЙНОЇ АТАКИ НА ШИФРИ СІМЕЙСТВА GRAIN ТА ЇХ ЗАГАЛЬНОЇ СТРУКТУРИ

Для подальшого аналізу шифру Grain потрібно більш детально дослідити його структуру. Для цього процесу зробимо спочатку загальний опис сімейства, а у наступному розділі розробимо міні версію шифру Grain-v0, для тестування на ній криптографічних атак. Також зробимо дослідження шивидкої кореляційної атаки та її застосування до шифру Grain-v1.

2.1 Загальна модель

Розділ створено на основі інформації з [2]. Опираючись на структуру шифрів Fruit, Sprout, різні версії шифру Grain а також інші схожі примітиви опишемо загальну модель Grain-подібних "легковагих"потокових шифрів.

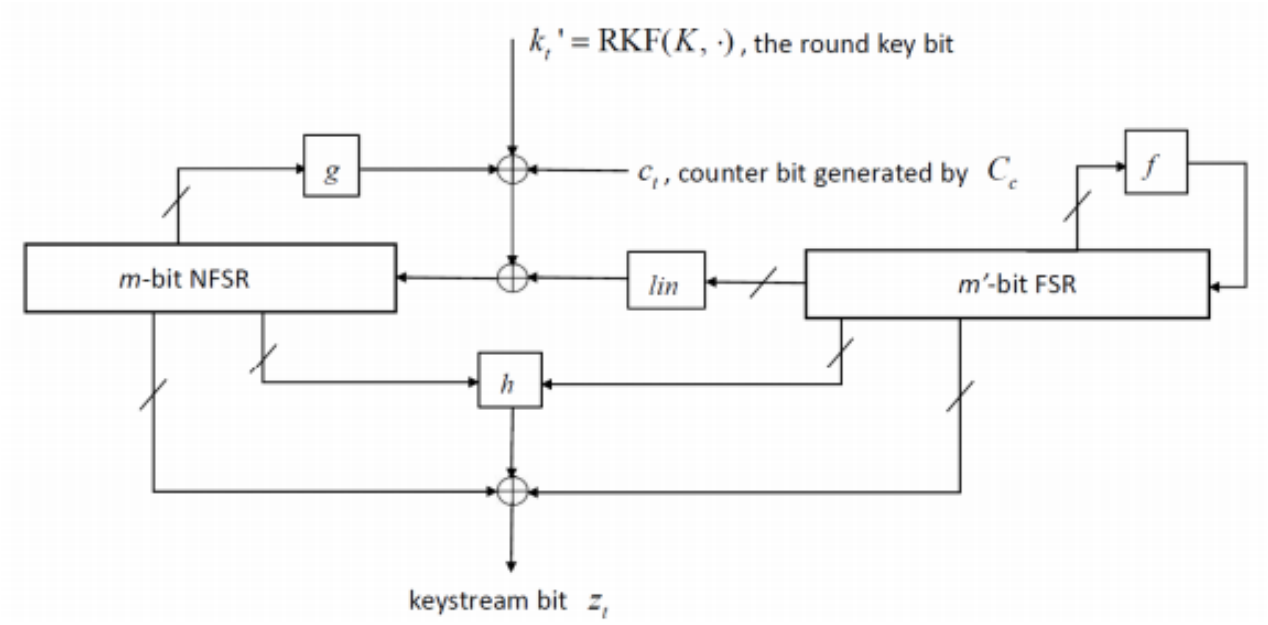

Рисунок 2.1 – Схема роботи Grain-подібних шифрів

Наступні позначення будуть використовуватись в данній моделі.

$N^t = (n_t,n_{t+1},...,n_{t+m-1})$ m - бітовий внутрішній стан РЗНЗЗ в момент часу t

$S^t = (s_t,s_{t+1},...,s_{t+m'-1})$ - m′ бітовий внутрішній стан РЗЗЗ в момент часу t, що оновлюється незалежно за допомогою лінійної або нелінійної функції.

$K = (k_0,k_1,...,k_{l-1})$ - l бітовий секретний ключ, що задовільнює умові $l \leqslant m + m^{'} \leqslant 2l$

$k_t^{'} = RKF(K, .)$ - біт раундового ключа, згенерований в момент часу t

C_c - раундовий лічильник для оновлення стану РЗНЗЗ

c_t - біт лічильника, згенерований C_c в момент часу t

$P_{S^t} = \{s_{t+\alpha_1},s_{t+\alpha_2},...,s_{t+\alpha_{j1}}\}$ - підмножина S^t та вхідні змінні функції фільтрації h з РЗЗЗ. $0 \leqslant \alpha_1 \leqslant \alpha_2... \leqslant \alpha_{j_1} \leqslant m^{'} - 1$

$P_{N^t} = \{n_{t+\beta_1},n_{t+\beta_2},...,n_{t+\beta_{j2}}\}$ - підмножина N^t та вхідні змінні функції фільтрації h з РЗНЗЗ. $0 \leqslant \beta_1 \leqslant \beta_2... \leqslant \beta_{j2} \leqslant m - 1$

$Q_{S^t} = \{s_{t+\sigma_1},s_{t+\sigma_2},...,s_{t+\sigma_{r1}}\}$ - підмножина S^t та вхідні змінні лінійної функції ϕ з FSR. $0 \leqslant \sigma_1 \leqslant \sigma_2... \leqslant \sigma_{r1} \leqslant m^{'} - 1$

$Q_{N^t} = \{n_{t+\eta_1},n_{t+\eta_2},...,n_{t+\eta_{r1}}\}$ - підмножина N^t та вхідні змінні лінійної функції ϕ з РЗНЗЗ. $0 \leqslant \eta_1 \leqslant \eta_2... \leqslant \eta_{r2} \leqslant m - 1$

Модель складається з п'яти булевих функцій: лінійна/нелінійна булева функція f, нелінійна функція g, лінійна функція lin, лінійна булева функція ψ та нелінійна функція фільтрації h.

На кожному кроці РЗЗЗ оновлюється незалежно за допомогою функції f, тоді як РЗНЗЗ оновлюється за допомогою g, раундового біта $k_t^{'}$, c_t та деяких бітів з РЗЗЗ. Біт $k_t^{'}$ в момент часу t генерується з RKF, що бере частину вхідних бітів з секретного ключа K.

Компоненти

Позначимо початковий стан РЗЗЗ S^0. Він обновлюється незалежно та рекурсивно за допомогою функції f. $S^{t+1} = (s_{t+1},s_{t+2},...,s_{t+m'})$, де $s_{t+m'} = f(S^t)$. Автори припускають, що цей процес зворотній. $S^{t-1} = (s_{t-1},s_t,...,s_{t+m'-2})$, де $s_{t-1} = f^{-1}(S^t)$

Позначимо початковий стан РЗНЗЗ N^0. Він обновлюється рекурсивно за допомогою лінійної функції g, а також елементів згенерованих за допомогою секретного ключа, лічильника C_c та РЗЗЗ. $n_{t+m} = k_t^{'} \oplus c_t \oplus lin(S^t) \oplus g(N^t)$. Аналогічно припускаємо, що цей процес зворотній. $n^{t-1} = k_{t-1}^{'} \oplus c_{t-1} \oplus lin^{'}(S^{t-1}) \oplus g^{-1}(N^t)$

Лінійна булева функція $\psi(.)$, що відображає з $GF(2)^{r1+r2}$ в $GF(2)$ використовується як частина вихідної функції. Та використовує $r1$ змінних зі стану РЗЗЗ, $r2$ змінних зі стану РЗНЗЗ.
$\psi_t \triangleq \psi(Q_{S^t},Q_{N^t}) = \Big(\bigoplus_{i=1}^{r_1} s_{t+\sigma_i} \Big) \oplus \Big(\bigoplus_{i=1}^{r_2} n_{t+\eta_i} \Big)$.

Функція фільтрації $h : GF(2)^{j1+j2} \rightarrow GF(2)$. $h_t \triangleq h(P_{S^t},P_{N^t})$ - використовується як інша частина вихідної функції, що використовує $j1$ вхідних значень з РЗЗЗ та $j2$ вхідних значень з РЗНЗЗ.

Вихідня функція $z(.) = h(.) \oplus \psi(.)$, що генерує потік $\{z_t\}_{t \geq 0}$

Властивості

· Передбачимо, що RKF періодична функція, тому раундові біти також. Припустимо, що p - найменше таке додатне ціле число $k_{t+p}^{'} = k_t^{'}$ для будь якого $t \geq 0$. Тобто раундові біти повторюються з періодичністю p. Будемо вважати, що c_t невідомий. В цьому випадку припустимо його періодичність. $c_{t+q} = c_t$ для будь якого $t \geq 0$

· Для функції фільтрації $h : GF(2)^{j1+j2} \rightarrow GF(2)$, $h_{P_{S^t}}(N^t)$, де $P_{S^t} \in GF(2)^{j_1}$ та $P_{N^t} \in GF(2)^{j_2}$ використовується для заміни $h(.)$ для фіксованого значення P_{S^t}. Припустимо, що для будь якого вибору P_{S^t}, $h_{P_{S^t}}$ буде лінійною булевою функцією в залежності від P_{N^t}. Зауважимо, що РЗЗЗ оновлюється незалежно, відповідно, для будь якого стану S^0 вихід моделі залежить лінійно від РЗНЗЗ. Звідси можемо інтерпертувати систему як лінійно фільтрований РЗНЗЗ, що включає в себе секретний раундовий біт з періодичністю p.

Розуміємо, що в нашій моделі РЗНЗЗ може бути розкладений на менші РЗНЗЗ, які також можуть бути атаковані. Очевидно, що модель описує Grain v1 та Fruit, але не Plantlet та Lizard. Розробники цих алгоритмів виключили властивість псевдолінійності, для захисту від швидкої кореляційної атаки.

Вхідними параметрами для шифру Grain v1 є $m = 80$, m' $= 80$ та $l = 80$, що є відповідно кількістю біт в РЗНЗЗ, РЗЛЗЗ та довжина ключа. До тих пір, поки ключ не використовується для генерації, $k_t^{'} = 0$ та $c_t^{'} = 0$. Вихідний біт генерується таким чином.

$z_t = h(s_{t+3},s_{t+25},s_{t+46},s_{t+64},s_{t+63}) \oplus n_{t+1} \oplus n_{t+2} \oplus n_{t+4} \oplus n_{t+10} \oplus n_{t+31} \oplus n_{t+43} \oplus n_{t+56}$, де

$h(.) = s_{t+25} \oplus \underline{n_{t+63}} \oplus s_{t+3}s_{t+64} \oplus s_{t+46}s_{t+64} \oplus s_{t+64}\underline{n_{t+63}} \oplus s_{t+3}s_{t+25}s_{t+46} \oplus s_{t+3}s_{t+46}s_{t+64} \oplus s_{t+3}s_{t+46}\underline{n_{t+63}} \oplus s_{t+25}s_{t+46}\underline{n_{t+63}} \oplus s_{t+46}s_{t+64}\underline{n_{t+63}}$

Підкреслені частини вказують на псевдолінійність функції $h(.)$

$P_{S^t} = \{s_{t+3},s_{t+25},s_{t+46},s_{t+64}\}$,

$Q_{S^t} = \varnothing$,

$P_{N^t} = \{n_{t+63}\}$,

$Q_{N^t} = \{n_{t+1},n_{t+2},n_{t+4},n_{t+10},n_{t+31},n_{t+43}\}$

Зауважимо, що $h_{P_{S^t}}$ - лінійна булева функція з входом з P_{N^t}, відповідно для будь якого стану РЗЛЗЗ вихід буде залежати від РЗНЗЗ. Довжина РЗЛЗЗ 80 біт, тож наша атака має складність 2^{80}, що робить її не ефективною.

Вхідними параметрами для шифру Fruit є $m = 37$, m' $= 43$ та $l = 80$, що є відповідно кількістю біт в РЗНЗЗ, РЗЛЗЗ та довжина ключа. В алгоритмі Fruit оновлення РЗНЗЗ залежить від секретного ключа. $k_t^{'} = k_{sv}k_{y+64} \oplus k_p k_{u+72} \oplus k_{q+32} \oplus k_{r+64} \triangleq RKF(K,C_r^t)$, де $C_r = (c_t^0,...,c_t^6)$. Зауважимо, що $k_t^{'}$ - періодична з $p = 128$. Також присутний лічильник $C_c = (c_t^7,...,c_t^{14})$ біт c_t^{10} якого присутній в оновленні РЗНЗЗ. Зауважимо, що лічильник C_c - відомий, та біт c_t^{10} має період $q = 32$. Значення які приймає біт c_t^{10} $\underbrace{0,...,0}_{16},\underbrace{1,...,1}_{16}$

Вихідний біт $z_t = h(s_{t+1},s_{t+6},s_{t+11},s_{t+15},s_{t+22}s_{t+27}s_{t+33},s_{t+42},n_{t+1},n_{t+33},n_{t+35}) \oplus$

$s_{t+38} \oplus n_t \oplus n_{t+7} \oplus n_{t+13} \oplus n_{t+19} \oplus n_{t+24} \oplus n_{t+29} \oplus n_{t+36}$

$h(.) = s_{t+15}\underline{n_{t+1}} \oplus s_{t+1}s_{t+22} \oplus s_{t+27}\underline{n_{t+35}} \oplus s_{t+11}\underline{n_{t+33}} \oplus s_{t+6}s_{t+33}s_{t+42}$, де підкресленим показано псевдолінійність.

$P_{S^t} = \{s_{t+1},s_{t+6},s_{t+11},s_{t+15},s_{t+22},s_{t+27},s_{t+33},s_{t+42}\}$,

$Q_{S^t} = \{s_{t+38}\}$,

$P_{N^t} = \{n_{t+1},n_{t+33},n_{t+35}\}$,

$Q_{N^t} = \{n_t,n_{t+7},n_{t+13},n_{t+19},n_{t+24},n_{t+29},n_{t+36}\}$.

$h_{P_{S^t}}$ - лінійна булева функція зі змінними $n_{t+1},n_{t+33},n_{t+35}$. Звісно для будь якого фіксованого значення РЗЛЗЗ, вихідні значення будуть лінійно залежати від РЗНЗЗ.

2.2 Кореляційний критпоаналіз шифру Grain

Швидка кореляційна атака - це доволі відомий класс криптографічних атак для потокових шифрів, що базуються на РЗЛЗЗ. Атака можлива за допомогою пошуку кореляцій між станом ЛЗРЗЗ та вихідною гаммою. Метою атаки є відновлення стану РЗЛЗЗ. У цьому розділі буде розглянуто атаку на Gain-v0 та його міні версію з нової точки зору, що вокористовує скінчені поля, а це в свою чергу вносить нові властивості для атаки, коли є багато лінійних наближень. До того ж буде використано новий алгоритм заснований на новій валстивості, що потребує значно менше часу та пам'яті. Атака буде використовуватись на шифрах сімейства Grain, що є добре проананлізованим сімейством шифрів. Наразі існує чотири потокових шифри із сімейства Grain-128a, Grain-128, Grain-v1 та Grain-v0.

2.3 Загальний опис кореляційної атаки

Розділ був створений за допомогою статті [3]. РЗЛЗЗ використовуються у потокових шифрах, що містять у собі декілька

РЗЛЗЗ та нелінійні функції. Для невеликого опису застосуємо рисунок 2.2

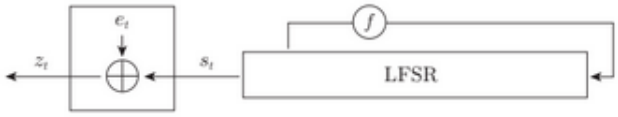

Рисунок 2.2 – Модель шифру, що заснована на РЗЛЗЗ

Швидкі кореляційні атаки є важливими атаками на потокові шифри, що базуються на РЗЛЗЗ. Основна ідея [15] була запропонована Зігенталером, що використовує розподіл бітів e_t. Припустимо, що внутрішній стан $S^0 = (s_0,s_1, ...,s_{n-1})$. Обчислюємо s_t, $t = n$, $n + 1$,...$N - 1$ та XORимо s_t з відповідним z_t. Якщо підібрано правильний початковий стан, то біт e_t підібрано правильно. З іншого боку, автори припускають, що XOR поводить себе випадково. Коли зібрано N бітів гамми та розмір регістру є n, то робота алгоритму займе $N2^n$. Для уникнення використання повного перебору, було запропоновано швидкі кореляційні атаки.

Основоположну роботу [16] було запропоновано Майером та Стафелбахом, в якій вони знайшли спосіб прибирати шум e_t з z_t за допомогою використання рівнянь перевірки парності, та відновили s_t. Також було запропоновано покарщення атаки, однак вони можливі лише на спрощені версії, де розмір РЗЛЗЗ менше, або відхилення шуму занадто велике, та вони не можуть бути використані на сучасних шифрах.

Іншим підходом до швидких кореляційних атак є використання так званих алгоритмів одного проходу, а він вже застосовний до сучасних алгоритмів. Аналогічно до звичайних кореляційних атак, вгадуємо початковий стан, та відновлюємо правильний за допомогою використання рівнянь перевірки парності. Щоб уникнути повного перебору початкового стану, було запропоновано декілька методів що зменшують кількість біт у початковому стані за допомогою використання рівнянь перевірки

парності. У найбільш успішних методах кількість використаних бітів зменшується за допомогою XORy двох різних рівнянь парності. Нехай $e_t = \langle\ s^{(0)}, a_t\ \rangle \oplus z_t$ рівняння парності, де $\langle\ s^{(0)}, a_t\ \rangle$ означає скалярний добуток між $s^{(0)}$, та a_t. Та припускаємо що e_t має велике відхилення. Без втрати загальності виявляємо множину пар (j_1, j_2), так що перші l бітів в $a_{j1} \oplus a_{j2}$ є нулем. Такий набір пар створюється за допомогою парадоксу днів народження. Далі $\langle\ s^0,\ \mathrm{a}_{j1} \oplus a_{j2}\rangle \oplus z_{j1} \oplus z_{j2}$ також має велике відхилення, та кількість використаних секретних бітів зменшується з n до $n - l$. Пізніше цей метод буде узагальнено за допомогою узагальненого парадоксу днів народження. Більш того, алгоритм було запропоновано для пришвидшення алгоритму одного проходу. Дослідники показали, що здогадка та оцінка може бути представлена як перетворення Уолша-Адамара, а швидке перетворення Уолша-Адамара може бути застосоване для пришвидшення алгоритму одного проходу. Коли наївний алгоритм виконується за $N2^n$, то швидке перетворення Уолша-Адамара виконується за $N + n2^n$. Коли кількість використаних біт зменшується з n до $n - l$, то кількість використаного часу зменшується до $N + (nl)2^{n-l}$. Недоліком використання алгоритму одного проходу є збільшення кількості шуму. Нехай p є ймовірністю того, що $e_t = 1$, а кореляція, що позначається c, де $c = 1 - 2p$. Якщо використовуємо XOR на рівняннях перевірки парності, щоб зменшити кількість використаних біт кореляція модифікованих рівнянь, падає до c^2. Підвищення шуму спричиняє підвищення скалдності виконання алгоритму.

2.4 Перегляд атаки швидкої кореляції

Розглянемо спочатку структуру рівнянь перевірки парності з іншої сторони, що буде полягати в використанні скінчених полів. Це надасть нових власитвостей швидкій кореляційній атаці. Множення між матрицями nxn та n бітовим фіксованим вектором зазвичай використовується для побудови рівнянь парності. Важливим є показати,

що це множення є комутативним над скінченними полями і це надає нових властивостей швидкій кореляційній атаці.

Спочатку розглянемо традиційну гіпотезу помилкового ключа. Тобто спостерігаємо кореляцію 0, коли бідібрано невірний початковий стан. Ця властивість означає, що нам потрібно переглянути гіпотезу невірного ключа більш детально. Зокрема, припускаючи, що існує декілька лінійних масок з великим відхиленням, що традиційна гіпотеза невірного ключа не використовує. Для цього існує модифікована гіпотеза. Нова властивість є корисною для підвищення ефективності швидкої кореляційної атаки, де є кілька лінійних масок з великим ухилом. В попередній швидкій кореляційній атаці наближення зменшували лише кількість використаної пам'яті, але не зменшували час роботи. Тому запропоновано алгоритм, що зменшує як час, так і кількість пам'яті, яку він буде використовувати. Новий алгоритм є одним з алгоритмів одного проходу, але спосіб уникання повного перебору сильно відрізняється від попереднього. Наразі використовуються лінійні маски для уникнення цього.

2.5 Порівняння попередніх атак на шифри сімейства Grain

Перед шифром Grain-v1 існує версія Grain-v0 і вона була зламана за допомогою швидкої кореляційної атаки. Шифр Grain-v1 побудований так, щоб прибрати вразливості шифру Grain-v0, однак за допомогою нової властивості нашої атаки можемо зламати шифр Grain-v1.

Near-collision атака є важливою атакою на шифр Grain-v1, та нещодавно було запропоновано поліпшення, fast near-collision атаку, автори якої стверджують, що часова складність досягє $2^{75.7}$. Однак ця оцінка є суперечливою, оскільки одиниця часової складності становить "1 функція оновлення довідкового коду для програмного забезпечення". Та вони порахували 1 функцію оновлення як $2^{10.4}$ циклів. Чиста часова складність є $2^{75.7+10.4} = 2^{86.1}$ тактів, що перевищує 2^{80}. З іншого боку

часова складність швидкої кореляційної атаки є $2^{76.7}$, де одиниця складності часу становить щонайбільше одне множення з фіксованими значеннями над кінцевим полем. Це очевидно швидше, аніж атака повного перебору, однак використовується більше пам'яті, аніж fast near-collision attack. Шифр Grain128 розроблявся більш стійким, аніж Grain-v1, де квадратична функція прийнята для нелінійного зворотнього зв'язку РЗНЗЗ. Нажаль такий низький степінь сприяє виникненню динамічної кубічної атаки. Робота Дінура і Шаміра була атакаю на слабкий ключ, пізніше вона була розширина до атаки одного ключа. Динамічна кубічна атака зламує ініціалізацію, а швидка кореляційна атака зламує генератор гамми. Зауважимо, що різні засоби використовуються для протидії атакам на генератор гамми, та ініціалізацію. Наприклад, можемо уникнути динамічної кубічної атаки за допомогою збільшення кількості раундів ініціалізації, однак цей засіб протидії не буде працювати для уникнення атаки на генератор гамми.

Шифр Grain-128a був створений щоб уникати динамічних кубічних атак. Степінь нелінійного поліному зворотнього зв'язку більша, ніж у Grain-128. У шифрі Grain-128a не було помічено недоліків у безпеці, однак інсують атаки, де кількість раундів ініціалізації зменшено.

2.6 Шифри, що базуються на РЗЛЗЗ

Ціллю швидкої кореляційної атаки є шифри, що базуються на РЗЛЗЗ, див. рисунок 3.1. Шифр РЗЛЗЗ генерує N-бітову вихідну послідовність $s_0,s_1,...,s_{N-1}$, а відповідною гаммою є $z_0,z_1,...,z_{N-1}$, що вираховується як $z_t = s_t \oplus e_t$, де e_t - двоічний шум. Нехай $f(x) = c_0 + c_1x^1 + c_2x^2 + ... + c_{n-1}x^{n-1} + x^n$ - поліном зворотнього зв'язку РЗЛЗЗ та $s^{(t)} = (s_t,s_{t+1},...,s_{t+n-1})$ - n - бітовий початковий стан РЗЛЗЗ в момент часу t. Потім на РЗЛЗЗ на виході має s_t і стан оновлюється до $s^{(t+1)}$.

Так що $s^{(t+1)} = s^{(t)} \times F = s^{(t)} \times \begin{pmatrix} 0 & . & . & . & 0 & 0 & c_0 \\ 1 & . & . & . & 0 & 0 & c_1 \\ . & . & . & . & . & . & . \\ 0 & . & . & . & 1 & 0 & c_{n-2} \\ 0 & . & . & . & 0 & 1 & c_{n-1} \end{pmatrix}$

Де F - двоїчна матриця розміром nxn , що представляє поліном зворотнього зв'язку $f(x)$. В данному РЗЛЗЗ шифрі, двоїчний шум e_t нелінійно згенеровано з початкового стану, або іншого початкового стану.

2.7 Швидкі кореляційні атаки

Швидкі кореляційні атаки використовують початковий стан РЗЛЗЗ та відповідною послідовністю гамми. Спочатку розглянемо найбільш просту модель, де приускаємо, що e_t саме по собі дуже відхилено. Нехай p буде ймовірністю того, що $e_t = 1$, а кореляція $c = 1 - 2p$. Пропонуємо початковий стан $s^{(0)}$, вираховуємо s_0,s_1,...,s_{N-1} з запропонованого $s^{(0)}$ та робимо оцінку $\sum_{t=0}^{N-1}(-1)^{s_t \oplus z_t}$, де сумма в результаті дасть ціле число. Якщо підібрано правильний початковий стан, то сумма буде дорівнювати $\sum_{t=0}^{N-1}(-1)^{e_t}$ та слідувати нормальному розподілу. N(N_c,N). З іншого боку, припускаємо, що сума себе поводить випадково, коли невірний початковий стан запропоновано. Тоді вона слідує нормальному розподілу N(0,N). Щоб розрізняти ці два розподіли, потрібно зібрати приблизно $N \approx (1/c^2)$ бітів з ключового потоку.

ШКА може бути розглянута як різновид лінійного криптоаналізу. Вихідний біт s_t лінійно вираховується з $s^{(0)}$. Як $s_t = \langle\ s^{(0)},\ A_t\ \rangle$, де A_t - перший вектор рядок транспонованої матриці F^t, що позначається як ${}^T F^t$. Іншими словами A_t використовується як лінійна маска, а метою

зловмисника є пошук $s^{(0)}$. Так що сума $\sum_{t=0}^{N-1}(-1)^{\langle s^{(0)},A_t\rangle}$ приймає такі значення, що далекі від $N/2$.

Зазвичай двоічний шум e_t має невелике відхилення в сучасних потокових шифрах, але можемо спостерігати високу кореляцію за допомогою додавання оптимально вибраних лінійних масок. Іншими словами, можемо використовувати ШПА, якщо

$$e_t^{'} = \bigoplus_{i\in T_s}\langle s^{t+i}, \Gamma_i\rangle \oplus \bigoplus_{i\in T_z} z_{t+i} \tag{2.1}$$

має велике відхилення за допомогою правильно підібраного T_s, T_z та Γ_i, а s^{t+i} Γ_i є n бітовими векторами. Згадаємо, що $s^{(t)} = s^{(0)} \times F^t$, з цього випливає, що

$$\begin{gathered} e_t^{'} = \bigoplus_{i\in T_s}\langle s^{t+i}, \Gamma_i\rangle \oplus \bigoplus_{i\in T_z} z_{t+i} = \\ \bigoplus_{i\in T_s}\langle s^{(0)} \times F^{t+i}, \Gamma_i\rangle \oplus \bigoplus_{i\in T_z} z_{t+i} = \\ \langle s^{(0)}, \left(\bigoplus_{i\in T_s}(\Gamma_i \times {}^T F^i)\right) \times {}^T F^t\rangle \oplus \bigoplus_{i\in T_z} z_{t+i}. \end{gathered} \tag{2.2}$$

Для спрощення використовуємо позначення Γ, що означає $\Gamma = \bigoplus_{i\in T_s}(\Gamma_i \times {}^T F^i)$. Наше рівняння перевірки парності може бути записано як

$$e_t^{'} = \langle s^{(0)},\Gamma \times {}^T F^t\rangle \oplus \bigoplus_{i\in T_z} z_{t+i} \tag{2.3}$$

Перевизначимо p як ймовірність того, що $e_t^{'} = 1$ для усіх можливих t, а кореляція c також перевизначеня для відповідного p. Далі можемо використовувати дане рівняння для ШКА. Припускаючи, що зібрано N рівнянь перевірки парності, спочатку пропонуємо $s^{(0)}$ та оцінюємо $\sum_{t=0}^{N-1}(-1)^{e_t^{'}}$. Допоки сума відповідає нормальному розподілу $\mathcal{N}(0,N)$ у випадковому випадку, сума буде відповідати $\mathcal{N}(N_c,N)$, якщо правильний початковий стан $s^{(0)}$ було підібрано.

Найбільш прямолінійний алгоритм має часову складність $\mathcal{O}(N2^n)$.

Процедура пропонування та оцінки може розглядатися як перетворення Уолша-Адамара. За допомогою швидкого перетворення Уолша-Адамара часова складність алгоритму може бути успішно застосована для прискорення алгоритму і його часова складність стає $\mathcal{O}(N + n2^n)$.

Означення 2.1 (Перетворення Уолша-Адамара)**.** $w : \{0,1\}^n \to \mathbb{Z}$ Перетворенням Уолша-Адамара функції w є

$$\hat{w}(s) = \sum_{x \in \{0,1\}^n} w(x)(-1)^{\langle s,x \rangle} \tag{2.4}$$

Коли вважаємо, що $s \in \{0,1\}^n$, емпірична кореляція

$$\sum_{t=0}^{N-1} (-1)^{e_t^{'}} \tag{2.5}$$

буде переписана як

$$\begin{aligned} \sum_{t=0}^{N-1} (-1)^{e_t^{'}} = \\ \sum_{t=0}^{N-1} (-1)^{\langle s^{(0)}, \Gamma \times^T F^t \rangle \oplus \bigoplus_{i \in T_z} z_{t+i}} = \\ \sum_{x \in \{0,1\}^n} \Big(\sum_{t \in \{0,\ldots,N-1 | \Gamma \times^T F^t = x\}} (-1)^{\langle s,x \rangle \oplus \bigoplus_{i \in T_z} z_{t+i}} \Big) = \\ \sum_{x \in \{0,1\}^n} \Big(\sum_{t \in \{0,\ldots,N-1 | \Gamma \times^T F^t = x\}} (-1)^{\bigoplus_{i \in T_z} z_{t+i}} \Big) (-1)^{\langle s,x \rangle} \end{aligned} \tag{2.6}$$

Звідси маємо, що

$$w(x) = \sum_{t \in \{0,\ldots,N-1 | \Gamma \times^T F^t = x\}} (-1)^{\bigoplus_{i \in T_z} z_{t+i}} \tag{2.7}$$

Наразі отримали $\hat{w}$ за допомогою швидкого перетворення Уолша-Адамара, де $\hat{w}$ - емпірична кореляція, при обраному s.

2.8 Перегляд швидкої кореляційної атаки

Для початку розглянемо структуру рівняння перевірки парності за допомогою використання скінчених полів, та покажемо, що $\Gamma \times {}^{T}F^{t}$ є комутативним. Це нове спостереження вносить нові властивості в ШКА і це є дуже важливим, коли є декілька лінійних масок. Як результат нам потрібно обережно розрізнити гіпотезу невірного ключа, наприклад, є випадки коли найбільш прості та типові гіпотези не існують. Більш того запропоновано алгоритм, що за допомогою використання нової властивості зменшує кількість використаної пам'яті та часову складність.

2.9 Огляд рівнянь перевірки парності за допомогою скінчених полів

Представимо $\Gamma \times {}^{T}F^{t}$ за допомогою використання $GF(2^n)$ де незвідний многочлен є поліномом зворотнього зв'язку регістру РЗЛЗЗ. Згадаємо, що $A_t \in \{0,1\}^n$ є першим рядком вектором у ${}^{T}F^{t}$, а потім iим рядком вектором у ${}^{T}F^{t}$, представленим як A_{t+i-1}. Нехай α буде таким елементом, що $f(\alpha) = 0$ і він буде примітивним елементом $GF(2^n)$. Цікавим спостереженням буде те, що $\Gamma \times {}^{T}F$ також стає природнім перетворенням $\gamma\alpha \in GF(2^n)$, тому що

$$\Gamma \times {}^{T}F = \Gamma \times \begin{pmatrix} 0\ 1\ .\ .\ 0\ 0 \\ .\ .\ .\ .\ .\ . \\ 0\ 0\ .\ .\ 1\ 0 \\ 0\ .\ .\ .\ 0\ 1 \\ c_0\ c_1\ .\ .\ c_{n-2}\ c_{n-1} \end{pmatrix}$$

З цього випливає, що $\Gamma \times {}^{T}F^{t}$ є також природнім перетворенням

$\gamma\alpha^t \in GF(2^n)$ а також множення є комутативним, наприклад $\gamma\alpha^t = \alpha^t\gamma$. Нарешті розглянемо матричне множення, що відповідає $\alpha^t\gamma$. Нехай M_γ буде двоїчною матрицею розміром nxn, де i й вектор рядок ${}^TM_\gamma$ визначений як природнє перетворення $\gamma\alpha^{i-1}$. Тоді $\alpha^t\gamma$ є природнім перетворенням $A_t \times {}^TM_\gamma$. Наведемо приклад для розуміння цього взаємозв'язку.

Приклад 2.1. $GF(2^8) = GF(2)[x]/(x^8 + x^4 + x^3 + x^2 + 1)$. Де $\Gamma = 01011011$. Транспонована матриця відповідної двоїчної матриці M_γ є

$$ {}^TM_\gamma = \begin{pmatrix} 0\,1\,0\,1\,1\,0\,1\,1 \\ 1\,0\,0\,1\,0\,1\,0\,1 \\ 1\,1\,1\,1\,0\,0\,1\,0 \\ 0\,1\,1\,1\,1\,0\,0\,1 \\ 1\,0\,0\,0\,0\,1\,0\,0 \\ 0\,1\,0\,0\,0\,0\,1\,0 \\ 0\,0\,1\,0\,0\,0\,0\,1 \\ 1\,0\,1\,0\,1\,0\,0\,0 \end{pmatrix} $$

Де перший рядок збігається з Γ, а другий рядок є природнім перетворенням $\gamma\alpha$. Тоді $\Gamma \times {}^TF^t = A_t \times {}^TM_\gamma$. Наприклад, коли $t = 10$,

$$\Gamma \times {}^{T}F^{10} = A_{10} \times {}^{T}M_{\gamma} \Longleftrightarrow (0\ 1\ 0\ 1\ 1\ 0\ 1\ 1) \times \begin{pmatrix} 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0 \\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0 \\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0 \\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0 \\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0 \\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0 \\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1 \\ 1\ 0\ 1\ 1\ 1\ 0\ 0\ 0 \end{pmatrix}^{10} =$$

$$= (0\ 0\ 1\ 0\ 1\ 1\ 1\ 0) \times \begin{pmatrix} 0\ 1\ 0\ 1\ 1\ 0\ 1\ 1 \\ 1\ 0\ 0\ 1\ 0\ 1\ 0\ 1 \\ 1\ 1\ 1\ 1\ 0\ 0\ 1\ 0 \\ 0\ 1\ 1\ 1\ 1\ 0\ 0\ 1 \\ 1\ 0\ 0\ 0\ 0\ 1\ 0\ 0 \\ 0\ 1\ 0\ 0\ 0\ 0\ 1\ 0 \\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 1 \\ 1\ 0\ 1\ 0\ 1\ 0\ 0\ 0 \end{pmatrix}$$ Як результат маємо 00010101.

Наразі можемо переписати

$$\langle s^{(0)}, \Gamma \times {}^{T}F^{t} \rangle = \langle s^{(0)}, A_t \times {}^{T}M_{\gamma} \rangle = s^{(0)} \times M_{\gamma}, A_t \tag{2.8}$$

, після таких перетворень перепишемо

$$e_t^{'} = M_{\gamma}, A_t \bigoplus_{i \in T_z} z_{t+i} \tag{2.9}$$

Використовуючи новий вигляд $e_t^{'}$ маємо таку властивість. Припускаємо, що можемо спостерігати високу кореляцію, коли пропонуємо $s^{(0)}$ та рівняння перевірки парності з $\Gamma \times {}^{T}F^{t}$. Тоді можемо спостерігати точно таку ж високу кореляцію, навіть якщо припускаємо $s^{(0)} \times M_{\gamma}$ та рівняння перевірки парності отримано з A_t замість $\Gamma \times {}^{T}F^{t}$. Надалі $\gamma \in GF(2^n)$ не відрізняється від $\Gamma \in \{0,1\}^n$ і використовуємо γ як лінійну маску для спрощення.

2.10 Нова гіпотеза невірного ключа

Автори зробили огляд традиційної гіпотези невірного ключа, що зазвичай використовується, у ній припустили, що емпірична кореляція поводить себе як випадкова, коли підібрано невірний початковий стан. Більш того з нової властивості випливає, що нам потрібно розглянути цю гіпотезу більш детально. Припускаємо, що використання лінійної маски Γ призводить до високої кореляції і називаємо такі лінійні маски лінійними масками з великим відхиленням. Коли генеруємо рівняння перевірки парності з $\Gamma \times {}^{T}F^{t}$ давайте розглянемо випадок, коли підібрали невірний початковий стан

$$s^{'(0)} = s^{(0)} \times M_{\gamma'} \tag{2.10}$$

З нової властивості випливає, що

$$\langle s^{'(0)}, \Gamma \times {}^{T}F^{t} \rangle = \langle s^{(0)} \times M_{\gamma'}, A_t \times {}^{T}M_{\gamma} \rangle = \langle S^{(0)}, A_t \times {}^{T}M_{\gamma\gamma'} \rangle \tag{2.11}$$

Іншими словами, це еквівалетно тому що $\gamma\gamma^{'}$ використовується як лінійна маска замість γ. Якщо $\gamma\gamma^{'}$ та γ є лінійними масками з великим відхиленням, то всеодно можемо спостерігати велику кореляцію, коли пропонуємо $s^{(0)} \times M_{\gamma'}$. Тому, припускаючи, що обраний потоковий шифр має кілька лінійних масок з високою кореляцією, вся відповідна здогадка приносить високу кореляцію.

Припустимо, що є m лінійних масок з високим відхиленням

γ_1,γ_2,...,γ_m та рівняння перевірки парності згенеровані з A_t. Далі спостерігаємо високу кореляцію, коли пропонуємо $s^{(0)} \times M_{\gamma_i}$ для будь-якого $i \in \{1,..,m\}$. В іншому випадку припускаємо, що воно поводить себе випадково, наприклад кореляція дорівнює нулю. Нова гіпотеза невірного ключа є розширенням старої гіпотези.

2.11 Новий алгоритм, що використовує нову властивість

Перед більш детальним аналізом атаки зробимо невеликий огляд. В цьому розділі n буде розміром РЗЛЗЗ у обраному РЗЛЗЗ побудованому шифрі. Припускаємо, що є $m(\ll 2^n)$ лінійних масок, що мають велике відхилення позначені як γ_1,..,γ_m. Процедура складається з трьох частин: побудови рівнянь перевірки парності, швидкого перетворення Уолша-Адамара, та видалення γ.

Спочатку будуємо рівняння перевірки парності. Рівняння перевірки парності традиційної швидкої кореляційної атаки побудовані з

$$\Gamma \times {}^T F^t \text{ та } \bigoplus_{i \in T_z} z_{t+i} \tag{2.12}$$

В нашому новому алгоритмі будуємо рівняння перевірки парності з A_t замість $\Gamma \times {}^T F^t$.

Далі використовуємо швидке перетворення Уолша-Адамара, щоб здобути розв'язок з високою кореляцією. Іншими словами, оцінюємо s таким чином, що

$$\langle s, A_t \rangle \oplus \bigoplus_{i \in T_z} z_{t+i} \tag{2.13}$$

має велике відхилення. Як було розглянуто раніше, будемо спостерігати високу кореляцію, коли $s = s^{(0)} \times M_{\gamma_i}$, та існує m розв'язків з високою кореляцією. Нажаль, навіть при застосуванні ШПУ, повинні запропонувати n бітів, а це вимагає $n2^n$ кроків. Це менш ефективно, аніж повний перебір, коли розмір РЗЛЗЗ є більшим або рівним порогу безпеки.

Щоб подолати цю проблему, будемо обходити деякі біти з n бітів за допомогою використання m лінійних масок. Зокрема обходимо β бітів, наприклад використовуємо лише $n - \beta$ бітів, а β бітів фіксовані, наприклад 0. Навіть якщо обійшли β бітів, маємо $m2^{-\beta}$ рішень з високою кореляцією, у середньому. Тому, $m2^{\beta}$ є необхідною умовою.

Далі обираємо рішення, у яких емпіричні кореляції більші за поріг, де деякі з рішень представлені як $s = s^{(0)} \times M_{\gamma_i}$. Щоб позбутися M_{γ_i} перебором бідбираємо γ_i та відновлюємо $s^{(0)}$. Припускаючи, що N_p рішень обрано часова складність становить $N_p \times m$. Якщо очікувана кількість подій, того що правильний стан $s^{(0)}$ з'явиться є значно більшою за кількість неправильно підібраних можемо легко знайти $s^{(0)}$. Симулюємо їх за допомогою розподілу Пуасона.

2.12 Більш детальний алгоритм

Нехай n буде розміром РЗЛЗЗ та k буде рівнем безпеки. Припускаємо, що існує m_p лінійних масок γ_1,...,γ_{m_p} с позитивною кореляцією, що більша за задане c. Більш того припускаємо, що є $m_m(\ll 2^n)$ лінійних масок p_1,p_2,...,p_{m_m} з негативною кореляцією, що менша за $-c$. Зауважимо, що c близько до нуля, а $m = m_m + m_p$.

Побудова рівнянь перевірки парності. Спочатку будуємо лінійні маски з A_t та $\bigoplus_{i \in T_z} z_{t+i}$ для $t = 0,1,..,N-1$, а часова складність дорівнює N. Емпірична кореляція відповідає $\mathcal{N}(N_c,N)$ та $\mathcal{N}(-N_c,N)$ коли пропонуємо один з $s^{(0)} \times M_{\gamma_i}$ та $s^{(0)} \times M_{\rho_i}$ відповідно. В іншому випадку припускаємо, що емпірична кореляція відповідає $\mathcal{N}(0,N)$.

Швидке перетворення Уолша-Адамара з обхідном спопобом. Далі обираємо $s \in \{0,1\}^n$ таке, що

$$\left| \frac{\sum_{t=0}^{N-1} (-1)^{e'_t}}{2} \right| \geq th \tag{2.14}$$

, де

$$e_t^{'} = \langle s,A_t \rangle \oplus \bigoplus_{i \in T_z} z_{t+i} \tag{2.15}$$

та $th(> 0)$ є порогом. Нехай $\epsilon 1$ - ймовірність того, що значення, які відповідають $\mathcal{N}(0,N)$ є більшими за th та нехай $\epsilon 2$ буде ймовірністю того, що значення які відповідають $\mathcal{N}(N_c,N)$ більші за th.

$$\epsilon 1 = \frac{1}{\sqrt{2\pi N}} \int_{th}^{\infty} \exp\left(-\frac{x^2}{2N}\right) dx \; \epsilon 2 = \frac{1}{\sqrt{2\pi N}} \int_{th}^{\infty} \exp\left(-\frac{(x-N_c)^2}{2N}\right) dx \tag{2.16}$$

Зауважимо, що ймовірність того, що значення які відповідають $\mathcal{N}(0,N)$ є меншими за $-th$ також дорівнює $\epsilon 1$, а ймовірність того, що значення які відповідають $\mathcal{N}(-N_c,N)$ є меншими за $-th$ є $\epsilon 2$. Нехай множини S_p та S_m є множинами обраних рішень з позитивними та негативними кореляціями відповідно. Очікуваний розмір S_p та S_m є $(2^n\epsilon 1 + m_p\epsilon 2)$ та $(2^n\epsilon 1 + m_m\epsilon 2)$, відповідно, коли вгадано ціле n бітове s. Нажаль, якщо вгадали ціле n бітове s, часова складність швидкого перетворення Уолша-Адамара є $n2^n$ і це є менш ефективним за повний перебір, коли $n \geq k$. Для того щоб зменшити часову складність пропонуємо декілька рішень. Замість вгадування повного s, вгадуємо його частину $(n - \beta)$ бітів, де пропущені біти стають константами, наприклад 0. Потім часова складність швидкого перетворення Уолша-Адамара зменшується з $n2^n$ до $(n - \beta)2^{n-\beta}$. Навіть якщо β бітів пропущено, $m_p 2^{-\beta}\epsilon 2$ рішень представлених як $s^{(0)} \times M_{\gamma_i}$ залишаються. Більш того розміри S_p та S_m зменшуються до $(2^{n-\beta}\epsilon 1 + m_p 2^{-\beta}\epsilon 2)$ та $(2^{n-\beta}\epsilon 1 + m_m 2^{-\beta}\epsilon 2)$ відповідно.

Видалення γ. Для всіх $s \in S_p$ та всіх $j \in \{1,2,..,m_p\}$ вираховуємо $s \times M_{\gamma_j}^{-1}$. Це дає $s^{(0)} \times M_{\gamma_i} \times M_{\gamma_j}^{-1}$ та становиться $s^{(0)}$, коли $i = j$. Оскільки є $m_p 2^{-\beta}\epsilon 2$ рішень представлених як $s^{(0)} \times M_{\gamma_i}$ в S_p, правильний $s^{(0)}$ з'являється $m_p 2^{-\beta}\epsilon 2$ разів. З іншого боку, кожен невірний початковий стан з'являється приблизно $m_p(2^{n-\beta}\epsilon 1) + m_p 2^{-\beta}\epsilon 2)2^{-n}$ разів коли припускаємо рівномірно випадкову поведінку. Кожен невірний

початковий стан з'являється приблизно

$$\begin{aligned} \lambda 1 = m_p(2^{n-\beta}\epsilon 1 + m_p 2^{-\beta}\epsilon 2)2^{-n} + \\ m_m(2^{n-\beta}\epsilon 1 + m_m 2^{-\beta}\epsilon 2)2^{-n} = \\ (m2^{n-\beta}\epsilon 1 + (m_p^2 + m_m^2)2^{-\beta}\epsilon 2)2^{-n} \end{aligned} \tag{2.17}$$

разів, коли припускаємо рівномірну випадкову поведінку. З іншого боку, правильний початковий стан $s^{(0)}$ з'являється

$$\lambda 2 = (m_p + m_m)2^{-\beta}\epsilon 2 = m2^{-\beta}\epsilon 2 \tag{2.18}$$

разів. Кількість випадків, коли з'вляється невірний початковий стан відповідає розподілу Пуасона за параметром $\lambda 1$, та кількість випадків, коли правильний $s^{(0)}$ з'являється відповідає розподілу Пуасона з параметром $\lambda 2$, щоб відновити унікальний початковий стан $s^{(0)}$ вводимо границю th_p наступним чином.

$$\sum_{k=th_p}^{\infty} \frac{\lambda 1^k e^{-\lambda 1}}{k!} < 2^{-n} \tag{2.19}$$

Ймовірність того, що кількість випадків, коли виникає $s^{(0)}$ є більшою за th_p оцінена як

$$\sum_{k=th_p}^{\infty} \frac{\lambda 2^k e^{-\lambda 2}}{k!} \tag{2.20}$$

Більш того, якщо ймовірність близька до 1, можемо однозначно відновити $s^{(0)}$ з високою ймовірністю.

2.13 Оцінка використаного часу та пам'яті

Процедура складається з трьох частин: побудови рівнянь перевірки парності, застосуванні швидкого перетворення Уолша-Адамара, та видалення γ. Перший крок вимагає часової складності N, де одиницею часової складності є множення за допомогою α над $GF(2^n)$ та $\bigoplus_{i \in T_z} z_{t+i}$.

Другий крок вимагає часової складності $(n-\beta)2^{n-\beta}$, де одиницею часової складності є додавання або віднимання. Останній крок вимагає часової складності

$$(m2^{n-\beta}\epsilon 1 + (m_m^2 + m_p^2)2^{-\beta}\epsilon 2) \tag{2.21}$$

, де одиницею часової складності є множення фіксованих значень над $GF(2^n)$. Ці одиниці часової складності не еквівалентні, але принаймні вони є більш ефективними, аніж одиниці, які дані ініціалізацією потокових шифрів. Тому для спрощення будемо вважати їх еквівалентними, та основная часова складність оцінюється як

$$N + (n-\beta)2^{n-\beta} + m2^{n-\beta}\epsilon 1 + (m_p^2 + m_m^2)2^{-\beta}\epsilon 2 \tag{2.22}$$

Нехай n буде розміром РЗЛЗЗ у РЗЛЗЗ побудованому потоковому шифрі. Припускаємо, що існує m лінійних масок, значення кореляції яких є більшим за c. Коли розмір пропущених бітів є β можемо відновити початковий стан РЗЛЗЗ з часовою складністю $3(n-\beta)2^{n-\beta}$ і необхідна кількість рівнянь перевірки парності є $N = (n-\beta)2^{(n-\beta)}$, де ймовірність успіху є

$$\sum_{k=th_p}^{\infty} \frac{\lambda 2^k e^{-\lambda 2}}{k!} \tag{2.23}$$

де th_p є мінімальним значенням, що задовілнює

$$\sum_{k=th_p}^{\infty} \frac{N^k e^{-N}}{k!} < 2^{-n} \tag{2.24}$$

та

$$\lambda 2 = \frac{m2^{-\beta}}{\sqrt{2\pi N}} \int_{th}^{\infty} \exp\left(-\frac{x - {N_c}^2}{2N}\right)dx, th = \sqrt{2N} \times erfc^{-1}\left(\frac{2(n-\beta)}{m}\right) \tag{2.25}$$

Кількість випадків, таких що кожне невірне значення з'являється має розподіл Пуассона з параметром $\lambda 1 = 2^{77.1498-80} = 2^{-2.8502}$. З іншого боку, кількість випадків коли з'являється $s^{(0)}$ мають розподіл Пуассона з

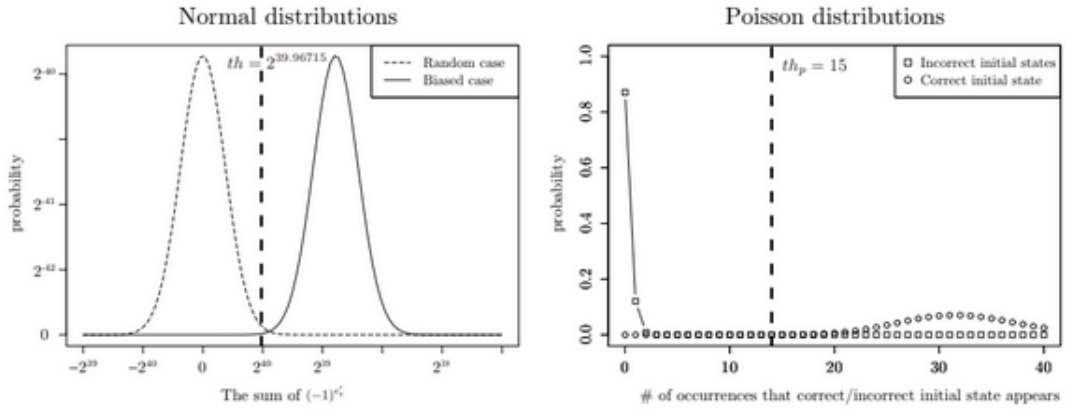

Рисунок 2.3 – Теоретична оцінка

параметром $\lambda 2 = 2^{14-9} \times 0.99957 \approx 31.98627$. На правому рисунку 2.3 зображено розподіл Пуассона. Наприклад, коли $th_p = 15$, ймовірність появи нового значення що найменше 15 разів є меншою, аніж 2^{-80}, однак відповідна ймовірність для $s^{(0)}$ є 99.9%. Як результат повною часовою складністю є $3 \times 2^{77.1498} \approx 2^{78.7348}$.

2.14 Застосування до Grain-v1

Застосуємо новий алгоритм до потокового шифру Grain-v1. Структура шифру Grain-v1 дуже схожа на структуру його попередника Grain-v0. Нехай $s^{(t)}$ та $b^{(t)}$ будуть 80 бітовими початковими станами РЗЛЗЗ та РЗНЗЗ в момент часу t відповідно. Вони представлені як $s^{(t)} = (s_t,..,s_{t+79})$ та $b^{(t)} = (b_t,..,b_{t+79)}$ відповідно.

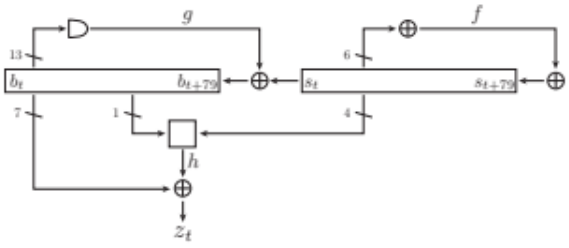

Рисунок 2.4 – Схема роботи шифру Grain-v1

Нехай z_t буде бітом, що генерується в момент часу t. Та він вираховується як

$$z_t = h(s^{(t)},b^{(t)}) \oplus \bigoplus_{j\in\mathcal{A}} b_{t+j} \tag{2.26}$$

де $A = \{1,2,4,10,31,43,56\}$ та $h(s^{(t)},b^{(t)})$ визначено як

$$\begin{aligned} h(s^{(t)},b^{(t)}) = \\ h(s_{t+3},s_{t+25},s_{t+46},s_{t+64},s_{t+63}) = \\ s_{t+25} \oplus b_{t+63}s_{t+3}s_{t+64} \oplus s_{t+46}s_{t+64} \oplus \\ s_{t+64}b_{t+63} \oplus s_{t+3}s_{t+25}s_{t+46} \oplus s_{t+3}s_{t+46}s_{t+64} \oplus \\ s_{t+3}s_{t+46}b_{t+63} \oplus s_{t+25}s_{t+46}b_{t+63} \oplus s_{t+46}s_{t+64}b_{t+63} \end{aligned} \tag{2.27}$$

Більш того біти s_{t+80} та b_{t+80} обраховуються наступним чином.

$$\begin{aligned} s_{t+80} = s_t \oplus s_{t+13} \oplus s_{t+23} \oplus s_{t+38} \oplus s_{t+51} \oplus s_{t+62} \\ b_{t+80} = s_t \oplus b_{t+62} \oplus b_{t+60} \oplus b_{t+52} \oplus b_{t+45} \oplus b_{t+37} \\ \oplus b_{t+33} \oplus b_{t+28} \oplus b_{t+21} \oplus b_{t+14} \oplus b_{t+9} \oplus b_t \oplus \\ b_{t+63}b_{t+60} \oplus b_{t+37}b_{t+33} \oplus b_{t+15}b_{t+9} \oplus b_{t+60}b_{t+52}b_{t+45} \\ \oplus b_{t+33}b_{t+28}b_{t+21} \oplus b_{t+63}b_{t+45}b_{t+28}b_{t+9} \\ \oplus b_{t+60}b_{t+52}b_{t+37}b_{t+33} \oplus b_{t+63}b_{t+60}b_{t+21}b_{t+15} \oplus \\ b_{t+63}b_{t+60}b_{t+52}b_{t+45}b_{t+37} \oplus b_{t+33}b_{t+28}b_{t+21}b_{t+15}b_{t+9} \oplus \\ b_{t+52}b_{t+45}b_{t+37}b_{t+33}b_{t+28}b_{t+21} \end{aligned} \tag{2.28}$$

Коли використовуємо $T_z = \{0,14,21,28,37,45,52,60,62,80\}$ звертаємо увагу на суму бітів ключового потоку. Наприклад $z_{t+0} \oplus z_{t+14} \oplus z_{t+21} \oplus z_{t+28} \oplus z_{t+37} \oplus z_{t+45} \oplus z_{t+52} \oplus z_{t+60} \oplus z_{t+60} \oplus z_{t+62} \oplus z_{t+80}$

$$\bigoplus_{i\in T_z} z_{t+i} = \bigoplus_{i\in T_z} h(s^{(t+i)},b^{(t+i)}) \oplus \bigoplus_{j\in A}(\bigoplus_{i\in T_z} b_{t+j+i}) \tag{2.29}$$

Для будь якого j

$$\bigoplus_{i \in T_z} b_{t+j+i} = s_{t+j} \oplus g^{'}(b^{(t+j)}) \tag{2.30}$$

де

$$\begin{aligned} g^{'}(b^{(t)}) = b_{t+33} \oplus b_{t+9} \oplus b_{t+63}b_{t+60} \oplus \\ b_{t+37}b_{t+33} \oplus b_{t+15}b_{t+9} \oplus b_{t+60}b_{t+52}b_{t+45} \\ \oplus b_{t+33}b_{t+28}b_{t+21} \oplus b_{t+63}b_{t+45}b_{t+28}b_{t+9} \\ \oplus b_{t+60}b_{t+52}b_{t+37}b_{t+33} \oplus \\ b_{t+63}b_{t+60}b_{t+21}b_{t+15} \oplus \\ b_{t+63}b_{t+60}b_{t+52}b_{t+45}b_{t+37} \oplus \\ b_{t+33}b_{t+28}b_{t+21}b_{t+15}b_{t+9} \oplus \\ b_{t+52}b_{t+45}b_{t+37}b_{t+33}b_{t+28}b_{t+21} \end{aligned} \tag{2.31}$$

Тоді

$$\begin{aligned} \bigoplus_{i \in T_z} z_{t+i} = \\ \bigoplus_{i \in T_z} h(s^{(t+i)},b^{(t+i)}) \oplus \bigoplus_{j \in A}(s_{t+j} \oplus g^{'}(b^{t+j})) = \\ \bigoplus_{j \in A} s_{t+j} \oplus \bigoplus_{i \in T_z} h(s^{(t+i)},b^{(t+i)}) \oplus \bigoplus_{j \in A} g^{'}(b^{(t+j)}) \end{aligned} \tag{2.32}$$

Далі розглянемо приблизне лінійне представлення $h(s^{(t+i)},b^{(t+i)})$. Нехай Λ_i буде вхідною лінійною маскою для функції h в момент часу $t+i$. Потім

$$h(s^{(t+i)},b^{(t+i)}) \approx \Lambda_i[4]b_{t+i+63} \oplus \langle \Lambda_i[0-3],(s_{t+i+3},s_{t+i+25},s_{t+i+46},s_{t+i+64}) \rangle \tag{2.33}$$

Нехай $cor_{h,i}(\Lambda_i)$ кореляція функції h в момент часу $t+i$, у таблиці показано результати. З таблиці видно, що $cor_{h,i}(\Lambda_i)$ є 0 або $\pm 2^{-2}$. Оскільки

маємо $|T_z| = 10$ активних функцій h, загальна кореляція усіх активних функцій h вираховується як

$$(-1)^{|T_z|+1} \prod_{i \in T_z} cor_{h,i}(\Lambda_i) = \pm 2^{-20} \tag{2.34}$$

Зауважимо, що $\Lambda_i[0-3]$ є незалежним від стану РЗНЗЗ. Усі терми, що використовуються у початковому стані РЗЛЗЗ можуть бути вгадані за допомогою ШКА.

Тому при кореляції $\pm 2^{-20}$ отримуємо

$$\bigoplus_{i \in T_z} z_{t+i} = (\text{терм обраний навмання}) \oplus \bigoplus_{i \in T_z} (\Lambda_i[4] b_{t+i+63}) \oplus \bigoplus_{j \in A} (g^{'}(b^{(t+j)})) \tag{2.35}$$

Тому, якщо

$$\begin{aligned} cor_g(\Lambda_{T_z}) = \\ Pr[\bigoplus_{i \in T_z} (\Lambda_i[4] b_{t+i+63}) \oplus \bigoplus_{j \in A} (g^{'}(b^{(t+j)})) = 0] - \\ Pr[\bigoplus_{i \in T_z} (\Lambda_i[4] b_{t+i+63}) \oplus \bigoplus_{j \in A} (g^{'}(b^{(t+j)})) = 1] \end{aligned} \tag{2.36}$$

є високою, то швидка кореляційна атака буде успішно застосована.

Аналогічно як у випадку з $Grain128a$ оцінюємо $cor_g(\Lambda_{T_z})$. Якщо одна з $\Lambda_0[4]$,$\Lambda_3 7[4]$,$\Lambda_5 2[4]$,$\Lambda_6 0[4]$,$\Lambda_6 2[4]$,та$\Lambda_8 0[4] \epsilon 1$, то кореляція завжди 0 оскільки b_{t+63},b_{t+100},b_{t+115},b_{t+123},b_{t+125} b_{t+143} не використані в $\bigoplus_{j \in A}(g^{'}(b^{(t+j)}))$. У таблиці показано, $cor_g(\Lambda_{T_z})$ коли $\Lambda_i[4] = 0$ для $i \in \{0,37,52,60,62,80\}$ Для будь-якого фіксованого Λ_i можемо підібрати лінійну апроксимацію

$$\bigoplus_{i \in T_z} z_{t+i} \approx \bigoplus_{j \in A} s_{t+j} \oplus \bigoplus_{i \in T_z} \langle \Lambda_i[0-3], (s_{t+i+3}, s_{t+i+25}, s_{t+i+46}, s_{t+i+64}) \rangle \tag{2.37}$$

Кореляція вираховується як $-cor_g(\Lambda_{T_z}) \times cor_h(\Lambda_{T_z})$

Пошук γ. Кореляція лінійного наближення за допомогою

$\Lambda_{14}[4]$	$\Lambda_{21}[4]$	$\Lambda_{28}[4]$	$\Lambda_{45}[4]$	$cor_g(\Lambda_{T_z})$
0	0	0	0	$-2^{-39.7159}$
0	0	0	1	$-2^{-43.4500}$
0	0	1	0	$-2^{-39.6603}$
0	0	1	1	$-2^{-43.7260}$
0	1	0	0	$+2^{-45.1228}$
0	1	0	1	$-2^{-42.9025}$
0	1	1	0	$+2^{-44.3802}$
0	1	1	1	$-2^{-42.6875}$
1	0	0	0	$+2^{-41.9519}$
1	0	0	1	$+2^{-43.5233}$
1	0	1	0	$+2^{-41.8662}$
1	0	1	1	$+2^{-43.6420}$
1	1	0	0	$-2^{-44.9114}$
1	1	0	1	$+2^{-42.8544}$
1	1	1	0	$-2^{-44.5232}$
1	1	1	1	$+2^{-42.7302}$

Рисунок 2.5 – Підсумок кореляцій, коли $\Lambda_i[4]$ є фіксованою

фіксованих Λ_i було оцінено у минулому параграфі. Лінійна маска γ, що використовується у ШКА виглядає наступним чином.

$$\gamma = \sum_{i \in T_z} (\Lambda_i[0]\alpha^{i+3} + \Lambda_i[1]\alpha^{i+25} + \Lambda_i[2]\alpha^{i+46} + \Lambda_i[3]\alpha^{i+64}) + \sum_{j \in A} \alpha^j \qquad (2.38)$$

Якщо різні Λ_h мають однакові γ нам потрібно скласти відподідні кореляції. Ця лінійна апроксимація не використовує $\Lambda_i[4]$ для $i \in T_z$. Більш того, треба просумувати $2^{|T_z|} = 2^{10}$ кореляцій, де $\Lambda_i[0-3]$ однакові та лише $A_i[5]$ варіюється для $i \in T_z$. Нехай V буде лінійною відстанню з базисом 12, відповідно до одиничних векторів.

Більш того, є спеціальні відносини, схожі як у Grain128a, та наразі маємо таких чотири.

$$\begin{aligned}
\Lambda_{37}[2] \text{ та } \Lambda_{80}[0]. \text{ Потім} \alpha^{37+46} = \alpha^{80+3} = \alpha^{83} \\
\Lambda_{62}[3] \text{ та } \Lambda_{80}[2]. \text{ Потім} \alpha^{62+64} = \alpha^{80+46} = \alpha^{126} \\
\Lambda_{0}[2] \text{ та } \Lambda_{21}[1]. \text{ Потім} \alpha^{0+46} = \alpha^{21+25} = \alpha^{46} \\
\Lambda_{2}1[3] \text{ та } \Lambda_{6}0[1]. \text{ Потім } \alpha^{21+64} = \alpha^{60+25} = \alpha^{85}
\end{aligned} \qquad (2.39)$$

Тому, з наступних чотирьох векторів

$$\begin{aligned} w1(\delta[0]) &= (00000,0^5,00000,0^5,00100,0^5,0^5,00000,00010,00\delta[1]00) \\ w2(\delta[1]) &= (00000,0^5,00000,0^5,00100,0^5,0^5,00000,00000,\delta[0]0000) \\ w3(\delta[2]) &= (00100,0^5,0\delta[2]000,0^5,00100,0^5,0^5,00000,00000,00000) \\ w4(\delta[3]) &= (00000,0^5,00010,0^5,00000,0^5,0^5,0\delta[3]000,00000,00000) \end{aligned} \tag{2.40}$$

Лінійну відстань $W(\delta) = span(w1(\delta[0]),w2(\delta[1],w3(\delta[2]),w4(\delta[3])))$ визначено, де $\delta[i]^{'} = \delta[i] \oplus 1$. Тоді нехай кореляцією γ буде cor_γ.

$$cor_\gamma = \sum_{w \in W(\delta)} \sum_{v \in V} -cor_g(\Lambda_{T_z} \oplus v) \times cor_h(\Lambda_{T_z} \oplus v \oplus w) \tag{2.41}$$

Евристично оцінили γ з високою кореляцією. Для кожного елементу T_z, оскільки підмножина $\{14,28,45,52\}$ незалежна від спеціальних відношень. Спочатку зосередимось на підмножині. Оскільки $b_{t+63+52}$ не використовується в $\bigoplus_{j \in A}(g^{'}(b^{(t+j)}))$. $\Lambda_5 2[4]$ повинна бути 0. Більш того $A_{52}[0-3]$ повинна бути обраною як
$A_{52}[0-3] \in \{0101,0111,1001,1011,1100,1101,1110,1111\}$, та кореляція γ є незмінною, оскільки використовуємо Λ_{52}, що задовільняє $cor_{h,52} = \pm 2^{-2}$. Нам більше не потрібно використовувати A_{52} і пошуковий простір зменшено з 2^{40} до 2^{36}.

Для $i \in \{14,28,45\}$ повинні бути обранівідповідні маски, а саме. $A_i[0-3] \in \{0101,0111,1001,1011,1100,1101,1110,1111\}$, оскільки $cor_g(\Lambda_{T_z})$ є високою коли $(\Lambda_1 4[4],\Lambda_2 1[4],\Lambda_{28}[4],\Lambda_{45}[4])$ є 0010 або 000. Наразі маємо три типи лінійних масок. $\Lambda_i[0-3] \in \{1001,1011,1100,1110\}$, де $cor_{h,i} = \pm 2^{-2}$ для $\Lambda_i[4] = 0$ але $cor_{h,i} = 0$ для $\Lambda_i[4] = 1$. $\Lambda_i[0-3] \in \{0111,1101\}$, де знак $cor_{h,i}$ відрізняється в обох випадках $\Lambda_i[4] = 0$ або 1. $\lambda_i[0-3] \in \{0101,1111\}$ де знак $cor_{h,i}$ є однаковим в обох випадках $\Lambda_i[4] = 0$ та 1. Оскільки cor_γ є незмінною в кожному випадку, то достатнім буде оцінити одну з кожного випадку. Більш того пошуковий простір зменшується з 2^{36} до $3^3 \times 2^{24}$

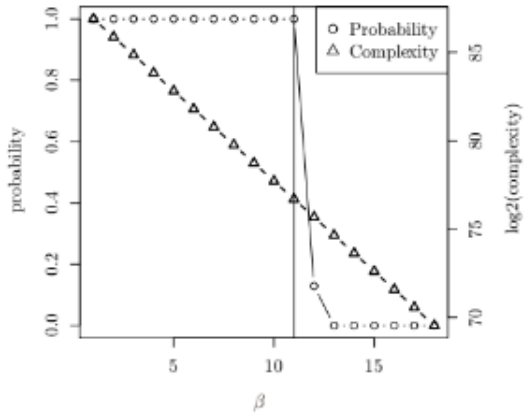

Рисунок 2.6 – Часова складність та ймовірність успіху. GrainV1

Висновки до розділу 2

У цьому розділі досліджено загальну структуру шифрів сімейства Grain. Зроблено загальний опис кореляційної атаки, переглянуто ататку швидкої кореляції. Порівняння попередніх атак на шифри сімейства Grain. Розглянуто нову гіпотезу невірного ключа. Проаналізовано наслідки нової властивості, а саме розглянуто новий алгоритм, що використовує нову властивість. А саме спочатку будується рівняння перевірки парності, потім використовується швидке перетворення Уолша-Адамара. Далі потрібно обрати рішення, у яких емпіричні кореляції більші за поріг. Розглянуто оцінку використаного часу та пам'яті. Проаналізовано застосування атаки до шифру Grain-v1.

3 МІНІ ВЕРСІЯ ШИФРУ GRAIN-V0

Для подальшого аналізу шифру Grain, пропоную міні версію шифру Grain-v0. Основною зміною буде зменшення кількості бітів у регістрах, в подальшому це дозволить більш ефективно перевіряти гіпотези щодо криптографічних атак, а також більш детально дослідити вже існуючу структуру.

3.1 Опис структури міні версії шифру Grain-v0

Наразі замість 80 біт у регістрах будемо використовувати 40 біт, це дозволить пришвидшити моменти потребуючі перебору, а також часу ініціалізації. Можна сказати, що масштаб шифру зменшився в два рази.

Оскільки розмір регістрів зменшився, то треба підібрати відповідний незвідний поліном та нову нелінійну функцію, а також обрати біти, що будуть входити в фукнцію фільтрації.

Новим поліномом зворотнього зв'язку для регістру РЗЛЗЗ буде [5]

$$f(x) = 1 + x^6 + x^7 + x^{18} + x^{28} + x^{36} + x^{40} \tag{3.1}$$

- незвідний поліном ступеня 40.

Для зворотнього зв'язку регістру РЗНЗЗ визначається новий поліном [4]

$$\begin{aligned} g(x) = 1 + x^{15} + x^{19} + x^{23} + x^{28} + x^{34} + x^{11}x^{24} + x^{20}x^{29} + x^{31}x^{37} + \\ x^{13}x^{18}x^{26} + x^{22}x^{28}x^{35} + x^{14}x^{24}x^{30}x^{37} + x^{18}x^{27}x^{31}x^{36} \end{aligned} \tag{3.2}$$

Для бітів регістру РЗЛЗЗ отримаємо вираз

$$s_{i+40} = s_{i+34} + s_{i+33} + s_{i+22} + s_{i+12} + s_{i+4} + s_i \tag{3.3}$$

А для бітів регістру РЗНЗЗ отримаємо вираз

$$b_{i+40} = s_i + b_{i+25} + b_{i+21} + b_{i+17} + b_{i+12} + b_{i+6} + b_{i+29}b_{i+16} + b_{i+20}b_{i+11} + b_{i+9}b_{i+3} + b_{i+27}b_{i+22}b_{i+14} + b_{i+18}b_{i+12}b_{i+5} + b_{i+26}b_{i+16}b_{i+10}b_{i+3} + b_{i+22}b_{i+13}b_{i+9}b_{i+4} \tag{3.4}$$

Зауважимо, що у виразі присутній біт s_i з регістру РЗЛЗЗ. Стан шифру описується вмістом регістрів зсуву. Як і в повній версії з регістрів обирається 5 бітів, які подаються на вхід булевої функції $h(x)$, яка є збалансованою, має кореляційний імунітет першого порядку, алгебраїчний степінь 3, та максимальну нелінійність. Функція $h(x)$ залишається такою ж як і в стандартній версії, а саме [1]

$$h(x) = x_1 + x_4 + x_0x_3 + x_2x_3 + x_3x_4 + x_0x_1x_2 + x_0x_2x_3 + x_0x_2x_4 + x_1x_2x_4 + x_2x_3x_4 \tag{3.5}$$

Однак змінні функції

$$x_0,\ x_1,\ x_2,\ x_3,\ x_4 \tag{3.6}$$

відповідають новим позиціям бітів

$$s_{i+3},\ s_{i+17},\ s_{i+29},\ s_{i+35} \text{ та } b_{i+25} \tag{3.7}$$

Результуючий біт маскується бітом b_i з РЗНЗЗ.

3.2 Ініціалізація та прискорення роботи міні версії

Перед початком роботи, шифр потрібно проініціалізувати. Зробимо припущення, та оберемо кількість тактів 80, однак можливо данна версія буде потребувати і білшу кількість, наприклад 160. Під час ініціалізації гамма не виробляється.

Перед початком роботи на вхід подаються ключ $K = k_0,...,k_{39}$ та вектор ініціалізації $IV = iv_0,...,iv_{31}$. Спочатку РЗНЗЗ заповнюється 40 бітовим ключем $b_i = k_i,\ 0 \leq i \leq 39$, потім заповнюємо перші 32 біти

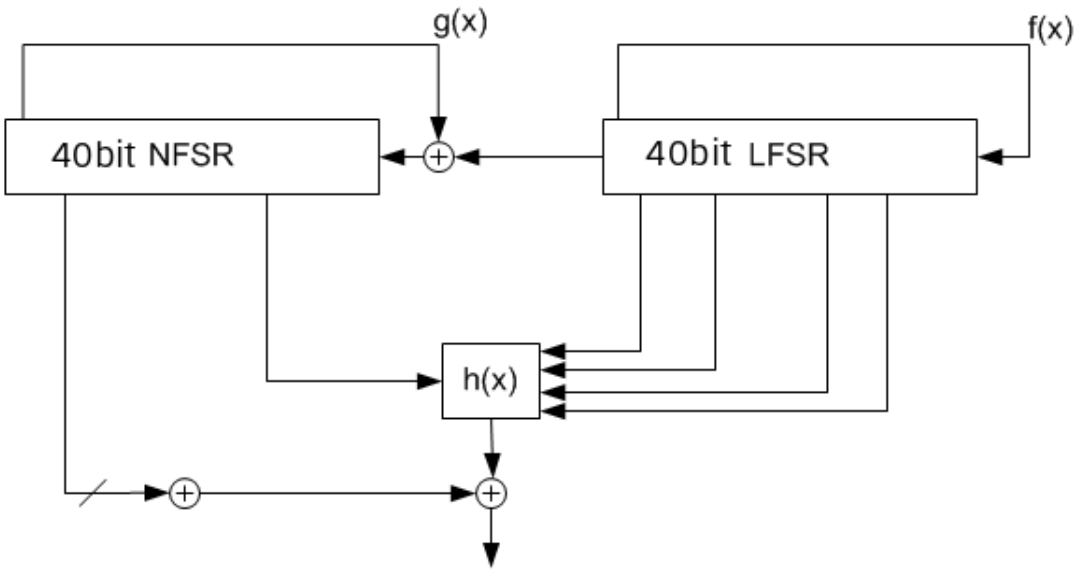

Рисунок 3.1 – Міні версія шифру Grain-v0

регістру РЗЛЗЗ вектором ініціалізації $s_i = iv_i, 0 \leq i \leq 31$, інші біти заповнюємо одиницями $s_i = 1, 32 \leq i \leq 39$. Після цього регістри не зможуть бути заповнені нулями.

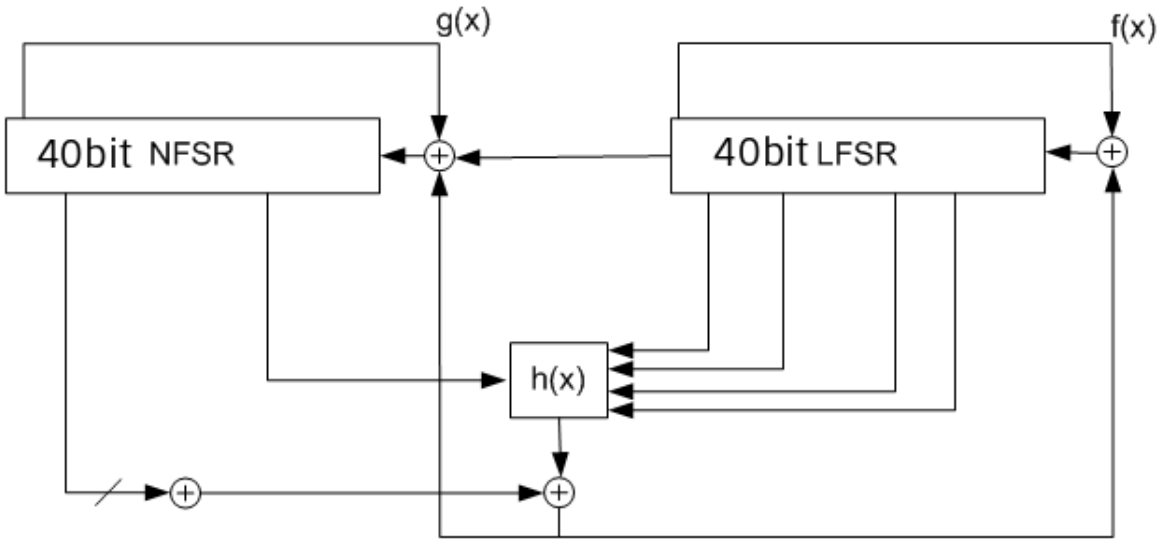

Рисунок 3.2 – Ініціалізація спрощеної версії шифру Grain

Як і в повній версії шифр можна прискорити за допомогою використання бітів, які не використовуються у функціях оновлення станів

$f(x)$, $g(x)$, та функції фільтрації $h(x)$. Такими бітами в данній версії є

$$s_i,\ 36 \leq i \leq 39 \tag{3.8}$$

та біти

$$b_i,\ 36 \leq i \leq 39 \tag{3.9}$$

Тобто маємо 4 позиції які можемо використовувати для одночасного використання функцій зворотнього зв'язку й функції фільтрації. З цього випливає, що можна пришвидшити роботу алгоритму до 4 разів. Якщо t - кількість біт які буде генерувати шифр за один такт,

$$t \leq 4, \text{ то } 80/t \tag{3.10}$$

- кількість тактів, які потрібно буде зробити для ініціалізації перед початком роботи.

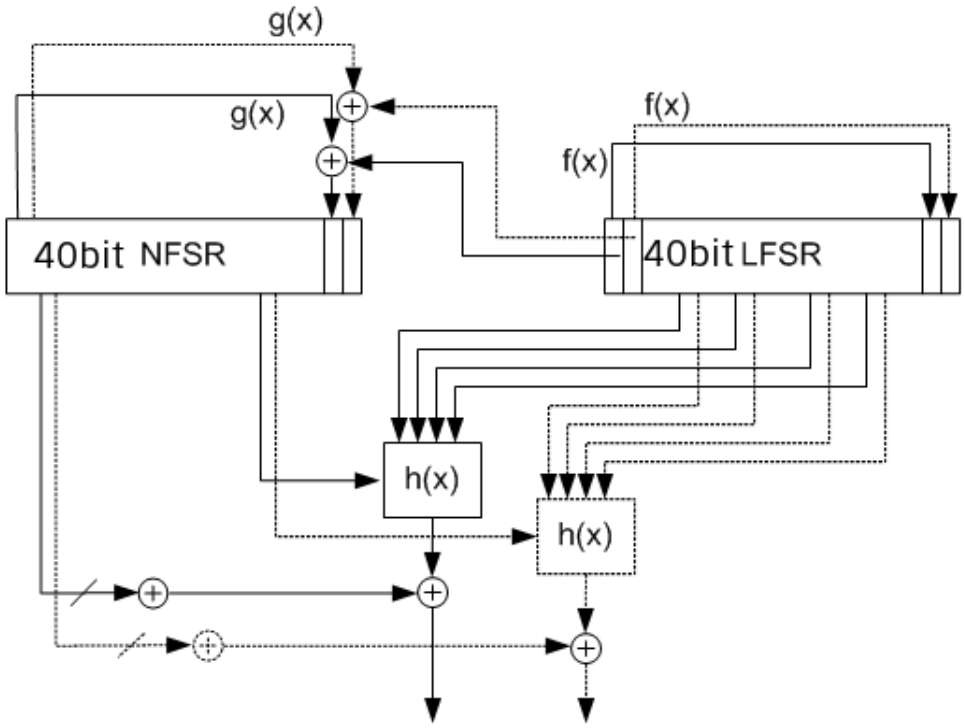

Рисунок 3.3 – Прискорення сроцеої версії шифру Grain

3.3 Швидкість роботи міні версії

Міні версія шифру Grain-v0 буде застосовуватись у дослідженні швидких кореляційних атак, які потребують певну кількість гамми. Для цього було зроблено програмну реалізацію шифру Grain-v0, а також міні версію Grain-v0 та заміри швидкості алгоритму.

Grain-v0		
Ключ 80 біт, Вектор 64 біт	Гамма 80 біт	Швидкість сек.
0x00000000000000000000 0x0000000000000000	0x5893bddf152dc7d5c26d	0.00175
0x10101010101010101010 0x1100110011001100	0x6e27e2bfc4e22eadb15e	0.00181
0x11111111111111111111 0x1111111111111111	0xfdee3076afb6819f069d	0.00195

mini Grain-v0		
Ключ 40 біт, Вектор 32 біт	Гамма 80 біт	Швидкість сек.
0x0000000000 0x00000000	0xe2d1c1a472eec13d6b82	0.00131
0x1010101010 0x10101010	0x4a49787aa4eeffb00080	0.00122
0x1111111111 0x11111111	0x87c6a8cd12490ebca9c9	0.00143

Як вже було згадано раніше, міні версія може бути прискорена у чотири рази, найбільш ефективної швидкості шифр набуває при апаратній реалізації.

3.4 Замітки щодо криптоаналізу міні версії

Враховуючи зміни у структурі шифру, критпоаналіз стає значно простішим. Наразі треба правильно враховувати положення деяких бітів, а в цілому плани атак залишаються як при повній версії. Оскільки на сьогодні актуальною криптографічною атакою є кореляційний криптоаналіз шифру, то подальше дослідження буде проводитись у цьому напрямку.

Висновки до розділу 3

У цьому розділі запропоновано міні версію ширфу Grain-v0, яка була розроблена таким чином, щоб зберегти властивості булевих функцій повної версії. А саме, повині зберігатись такі властивості: збалансованість, кореляційний імунітет, алгебраїчний степінь, нелінійність. Оскільки шифр Grain-v0 є масштабованим, то він був зменшений в два рази. Наразі він складається з двох регістрів зсуву зі зворотним зв'язком, кожен з яких містить по 40 біт. Лівий регістр є регістром зсуву з нелінійним зворотнім зв'язком, а правий має лінійний зворотній зв'язок. Перед початком роботи шифр ініціалізується, а вже потім генерує гамму. Міні версія також може бути прискорена, а саме у чотири рази.

ВИСНОВКИ

У результаті цієї роботи були проаналізовані та стисло описані джерела за тематикою дослідження, а саме шифр Grain-v0 та огляд вже існуючих криптографічних атак на шифри сімейства Grain.

Зроблено огляд загальної структури шифрів сімейства Grain, досліджено базові компоненти шифрів сімейства, та їх взаємозв'язок.

Досліджено вже існуючу кореляційну атаку, зроблено її загальний опис. Також її було переглянуто та порівняно з попередніми атаками. Розглянуто швидкі кореляційні атаки, а також їх перегляд. Зроблено огляд рівнянь перевірки парності за допомогою скінчених полів. Досліджено нову гіпотезу невірного ключа, а також новий алгоритм, що використовує нові властивості. Показано застосування швидкої кореляційної атаки до шифру Grain-v1.

Розроблено міні версію шифру Grain-v0, регістри якої зберігають властивості регістрів шифру Grain-v0. Тобто повна версія шифру є машсштабованою. Міні версія буде використовуватись в подальших дослідженнях криптографічних атак.

Зокрема представлена практична реалізація шифру Grain-v0 та його міні версії. Зроблено заміри швидкості обох шифрів.

У подальшій роботі планується створити модель швидкої кореляційної атаки на шифр Grain-v0 та його міні версію. Що дозволить далі вивчати стійкіть шифрів сімейства Grain до швидких кореляційних атак.

ПЕРЕЛІК ПОСИЛАНЬ

1. Hell, Martin, Thomas Johansson, and Willi Meier. *Grain: a stream cipher for constrained environments.* IJWMC 2.1 (2007): 86-93.

2. Zhang, Bin, Xinxin Gong, and Willi Meier. *Fast correlation attacks on Grain-like small state stream ciphers.* IACR Transactions on Symmetric Cryptology (2017): 58-81.

3. Todo, Yosuke, et al. *Fast correlation attack revisited.* Annual International Cryptology Conference. Springer, Cham, 2018.

4. Ding, Lina, et al. *A new lightweight stream cipher based on chaos.* Symmetry 11.7 (2019): 853.

5. Živković, Miodrag. *A table of primitive binary polynomials.* Mathematics of Computation 62.205 (1994): 385-386.

6. Todo, Yosuke, Willi Meier, and Kazumaro Aoki. *On the Data Limitation of Small-State Stream Ciphers: Correlation Attacks on Fruit-80 and Plantlet.* International Conference on Selected Areas in Cryptography. Springer, Cham, 2019.

7. Berbain, Côme, Henri Gilbert, and Antoine Joux. *Algebraic and correlation attacks against linearly filtered non linear feedback shift registers."International Workshop on Selected Areas in Cryptography. Springer, Berlin, Heidelberg, 2008.*

8. Watson, E. J. *Primitive polynomials (mod 2).* Math. Comp 16.368 (1962): 1962.

9. Bogdanov, Andrey, et al. *PRESENT: An ultra-lightweight block cipher.* International Workshop on Cryptographic Hardware and Embedded Systems. Springer, Berlin, Heidelberg, 2007.

10. Dey, Sabyasachi, and Santanu Sarkar. *Cryptanalysis of full round Fruit.* IACR Cryptology ePrint Archive 2017 (2017): 87.

11. Zhang, Bin, and Dengguo Feng. *Multi-pass fast correlation attack on stream ciphers.* International Workshop on Selected Areas in Cryptography.

Springer, Berlin, Heidelberg, 2006.

12. Chepyzhov, Vladimor V., Thomas Johansson, and Ben Smeets. *A simple algorithm for fast correlation attacks on stream ciphers.* International Workshop on Fast Software Encryption. Springer, Berlin, Heidelberg, 2000.

13. Chose, Philippe, Antoine Joux, and Michel Mitton. *Fast correlation attacks: An algorithmic point of view.* International Conference on the Theory and Applications of Cryptographic Techniques. Springer, Berlin, Heidelberg, 2002.

14. Hell, Martin, et al. *Grain-128AEAD-A lightweight AEAD stream cipher.* NIST Lightweight Cryptography, Round 1.

15. Siegenthaler, Thomas. *Decrypting a class of stream ciphers using ciphertext only.* IEEE Transactions on computers 1 (1985): 81-85.

16. Meier, Willi, and Othmar Staffelbach. *Fast correlation attacks on certain stream ciphers.* Journal of cryptology 1.3 (1989): 159-176.

ДОДАТОК А ТЕКСТИ ПРОГРАМ

Програмна реалізація шифру Grain-v0.

А.1 Програма 1

```
import numpy as np
from timeit import default_timer as timer

# main blocks Grain-v0
lfsr = np.zeros(80,dtype=bool)
nfsr = np.zeros(80,dtype=bool)

# conversion
binary = lambda x: "".join(reversed( [i+j
for i,j in zip( *[ ["{0:04b}".format(int(c,16))
for c in reversed("0"+x)][n::2] for n in [1,0] ] ) ] ))

# initialization of nfsr and lfsr with Key and IV
# fill nfsr with key bits, bi = ki, i from 0 to 79
# fill first 64 bits of lfsr with initialization vector bits,
#  i from 0 to 63
# rest of lfsr fill with ones, si = 1, i from 64 to 79
# in case of this action lfsr cannot be full of zeros
def initialization(key,iv):
    iv_bin = binary(iv)
    lfsr[:64] = [bool( int(iv_bin[ix])) for ix in range(64)]
    lfsr[64:] = 1
    key_bin = binary(key)
```

```
    nfsr[:80] = [bool( int(key_bin[ix])) for ix in range(80)]

def rounds():
    h_x=0
    f_x=0
    g_x=0

    for ix in range(160):
        f_x = lfsr[62] ^ lfsr[51] ^ lfsr[38] ^ lfsr[23] ^
              lfsr[13] ^ lfsr[0] ^ h_x
        g_x = lfsr[0]  ^ nfsr[63] ^ nfsr[60] ^ nfsr[52] ^
              nfsr[45] ^ nfsr[37] ^ nfsr[33] ^ nfsr[28] ^
              nfsr[21] ^ nfsr[15] ^ nfsr[9] ^ nfsr[0] ^
              (nfsr[63] & nfsr[60]) ^ (nfsr[37] & nfsr[33]) ^
              (nfsr[15] & nfsr[9]) ^ (nfsr[60] & nfsr[52] &
              nfsr[45]) ^ (nfsr[33] & nfsr[28] & nfsr[21]) ^
              (nfsr[63] & nfsr[45] & nfsr[28] & nfsr[9]) ^
              (nfsr[60] & nfsr[52] & nfsr[37] & nfsr[33]) ^
              (nfsr[63] & nfsr[60] & nfsr[21] & nfsr[15]) ^
              (nfsr[63] & nfsr[60] & nfsr[52] & nfsr[45] &
              nfsr[37]) ^ (nfsr[33] & nfsr[28] & nfsr[21] &
              nfsr[15] & nfsr[9]) ^ (nfsr[52] & nfsr[45] &
              nfsr[37] & nfsr[33] & nfsr[28] & nfsr[21]) ^
              h_x

        x0 = lfsr[0]
        x1 = lfsr[25]
        x2 = lfsr[46]
        x3 = lfsr[64]
        x4 = nfsr[63]
        h_x = x1 ^ x4 ^ (x0 & x3) ^ (x2 & x3) ^ (x3 & x4) ^
```

```
        (x0 & x1 & x2) ^ (x0 & x2 & x3) ^ (x0 & x2 & x4) ^
        (x1 & x2 & x4) ^ (x2 & x3 & x4)
        h_x = h_x ^ nfsr[0]

        lfsr[:-1] = lfsr[1:] #left shift
        lfsr[-1] = f_x
        nfsr[:-1] = nfsr[1:] #left shift
        nfsr[-1] = g_x

#gamma generator
def stream():
    h_x = 0
    res = np.zeros(80,dtype=bool)
    for ix in range(80):
        f_x = lfsr[62] ^ lfsr[51] ^ lfsr[38] ^ lfsr[23] ^
        lfsr[13] ^ lfsr[0]
        g_x = lfsr[0] ^ nfsr[63] ^ nfsr[60] ^ nfsr[52] ^
              nfsr[45] ^ nfsr[37] ^ nfsr[33] ^ nfsr[28] ^
              nfsr[21] ^ nfsr[15] ^ nfsr[9] ^ nfsr[0] ^
              (nfsr[63] & nfsr[60]) ^ (nfsr[37] & nfsr[33]) ^
              (nfsr[15] & nfsr[9]) ^ (nfsr[60] & nfsr[52] &
              nfsr[45]) ^ (nfsr[33] & nfsr[28] & nfsr[21]) ^
              (nfsr[63] &nfsr[45] & nfsr[28] & nfsr[9]) ^
              (nfsr[60] & nfsr[52] & nfsr[37] & nfsr[33]) ^
              (nfsr[63] & nfsr[60] & nfsr[21] & nfsr[15]) ^
              (nfsr[63] & nfsr[60] & nfsr[52] & nfsr[45] &
              nfsr[37]) ^ (nfsr[33] & nfsr[28] & nfsr[21] &
              nfsr[15] & nfsr[9]) ^ (nfsr[52] & nfsr[45] &
              nfsr[37] & nfsr[33] & nfsr[28] & nfsr[21])

        x0 = lfsr[0]
```

```
        x1 = lfsr[25]
        x2 = lfsr[46]
        x3 = lfsr[64]
        x4 = nfsr[63]
        h_x = x1 ^ x4 ^ (x0 & x3) ^ (x2 & x3) ^ (x3 & x4) ^
        (x0 & x1 & x2) ^ (x0 & x2 & x3) ^ (x0 & x2 & x4) ^
        (x1 & x2 & x4) ^ (x2 & x3 & x4)
        h_x = h_x ^ nfsr[0]

        lfsr[:-1] = lfsr[1:] #
        lfsr[-1] = f_x
        nfsr[:-1] = nfsr[1:] #
        nfsr[-1] = g_x
        res[ix] = h_x
    return res

def Grain_v0(key, iv):
    initialization( key, iv)
    rounds()
    key_stream = stream()
    result = ""
    for i in key_stream:
        if i == True:
            result += "1"
        else:
            result += "0"

    hstr = '%0*x' % ((len(result) + 3) // 4, int(result, 2))
    print(hstr)
    lfsr = np.zeros(80,dtype=bool)
    nfsr = np.zeros(80,dtype=bool)
```

```
start = timer()
Grain_v0("00000000000000000000", "0000000000000000")
end = timer()
print(end - start)

start = timer()
Grain_v0("10101010101010101010", "1100110011001100")
end = timer()
print(end - start)

start = timer()
Grain_v0("11111111111111111111", "1111111111111111")
end = timer()
print(end - start)
```

Програмна реалізація міні версії шифру Grain-v0.

A.2 Програма 2

```
import numpy as np
from timeit import default_timer as timer

# main blocks mini Grain-v0
lfsr = np.zeros(40,dtype=bool)
nfsr = np.zeros(40,dtype=bool)

# conversion
binary = lambda x: "".join(reversed( [i+j
for i,j in zip( *[ ["{0:04b}".format(int(c,16))
for c in reversed("0"+x)][n::2] for n in [1,0] ] ) ] ))
```

```
# initialization of nfsr and lfsr with Key and IV
# fill nfsr with key bits, bi = ki, i from 0 to 39
# fill first 32 bits of lfsr with initialization vector bits,
# i from 0 to 63
# rest of lfsr fill with ones, si = 1, i from 32 to 39
# in case of this action lfsr cannot be full of zeros
def initialization(key,iv):
    iv_bin = binary(iv)
    lfsr[:32] = [bool( int(iv_bin[ix])) for ix in range(32)]
    lfsr[32:] = 1
    key_bin = binary(key)
    nfsr[:40] = [bool( int(key_bin[ix])) for ix in range(40)]

def rounds():
    h_x=0
    f_x=0
    g_x=0

    for ix in range(80):
        f_x = lfsr[34] ^ lfsr[33] ^ lfsr[22] ^ lfsr[12] ^
        lfsr[4] ^ lfsr[0] ^ h_x
        g_x = lfsr[0] ^ nfsr[25] ^ nfsr[21] ^ nfsr[17] ^
        nfsr[12] ^ nfsr[6] ^ (nfsr[29] & nfsr[16]) ^
        (nfsr[20] & nfsr[11]) ^ (nfsr[9] & nfsr[3]) +
        (nfsr[27] & nfsr[22] & nfsr[14]) ^
        (nfsr[18] & nfsr[12] & nfsr[5]) ^
        (nfsr[26] & nfsr[16] & nfsr[10] & nfsr[3]) ^
        (nfsr[22] & nfsr[13] & nfsr[9] & nfsr[4]) ^ h_x

        x0 = lfsr[3]
        x1 = lfsr[17]
```

```
        x2 = lfsr[29]
        x3 = lfsr[35]
        x4 = nfsr[25]
        h_x = x1 ^ x4 ^ (x0 & x3) ^ (x2 & x3) ^
        (x3 & x4) ^ (x0 & x1 & x2) ^ (x0 & x2 & x3) ^
        (x0 & x2 & x4) ^ (x1 & x2 & x4) ^
        (x2 & x3 & x4)
        h_x = h_x ^ nfsr[0]

        lfsr[:-1] = lfsr[1:] #
        lfsr[-1] = f_x
        nfsr[:-1] = nfsr[1:] #
        nfsr[-1] = g_x

#
def stream():
    h_x = 0
    res = np.zeros(80,dtype=bool)
    for ix in range(80):
        f_x = lfsr[34] ^ lfsr[33] ^ lfsr[22] ^ lfsr[12] ^
        lfsr[4] ^ lfsr[0]
        g_x = lfsr[0] ^ nfsr[25] ^ nfsr[21] ^ nfsr[17] ^
        nfsr[12] ^ nfsr[6] ^ (nfsr[29] & nfsr[16]) ^
        (nfsr[20] & nfsr[11]) ^ (nfsr[9] & nfsr[3]) +
        (nfsr[27] & nfsr[22] & nfsr[14]) ^
        (nfsr[18] & nfsr[12] & nfsr[5]) ^
        (nfsr[26] & nfsr[16] & nfsr[10] & nfsr[3]) ^
        (nfsr[22] & nfsr[13] & nfsr[9] & nfsr[4])

        x0 = lfsr[3]
        x1 = lfsr[17]
```

```
        x2 = lfsr[29]
        x3 = lfsr[35]
        x4 = nfsr[25]
        h_x = x1 ^ x4 ^ (x0 & x3) ^ (x2 & x3) ^
        (x3 & x4) ^ (x0 & x1 & x2) ^ (x0 & x2 & x3) ^
        (x0 & x2 & x4) ^ (x1 & x2 & x4) ^ (x2 & x3 & x4)
        h_x = h_x ^ nfsr[0]

        lfsr[:-1] = lfsr[1:] #
        lfsr[-1] = f_x
        nfsr[:-1] = nfsr[1:] #
        nfsr[-1] = g_x
        res[ix] = h_x
    return res

def Grain_v0_mini(key, iv):
    initialization( key, iv)
    rounds()
    key_stream = stream()
    result = ""
    for i in key_stream:
        if i == True:
            result += "1"
        else:
            result += "0"

    hstr = '%0*x' % ((len(result) + 3) // 4, int(result, 2))
    print(hstr)
    lfsr = np.zeros(80,dtype=bool)
    nfsr = np.zeros(80,dtype=bool)
```

```
start = timer()
Grain_v0_mini("0000000000", "00000000")
end = timer()
print(end - start)

start = timer()
Grain_v0_mini("1010101010", "10101010")
end = timer()
print(end - start)

start = timer()
Grain_v0_mini("1111111111", "11111111")
end = timer()
print(end - start)
```